The 19th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision

in co-operation with

EUROGRAPHICS

W S C G ' 2011

Communication Papers Proceedings

University of West Bohemia
Plzen
Czech Republic

January 31 – February 3, 2011

Co-Chairs
Gladimir Baranoski, University of Waterloo, Canada
Vaclav Skala, University of West Bohemia, Czech Republic

Edited by
Gladimir Baranoski, Vaclav Skala

Vaclav Skala – Union Agency

WSCG'2011 Communication Papers Proceedings

Editor-in-Chief: Vaclav Skala
c/o University of West Bohemia, Univerzitni 8
CZ 306 14 Plzen
Czech Republic
skala@kiv.zcu.cz

Managing Editor: Vaclav Skala

Published and printed by:
Vaclav Skala – Union Agency
Na Mazinách 9
CZ 322 00 Plzen
Czech Republic

Hardcopy: *ISBN 978-80-86943-82-4*

WSCG 2011

International Program Committee

Balcisoy, S. (Turkey)
Baranoski, G. (Canada)
Benes, B. (United States)
Benoit, C. (France)
Bilbao, J. (Spain)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bouatouch, K. (France)
Buehler, K. (Austria)
Coquillart, S. (France)
Daniel, M. (France)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Feito, F. (Spain)
Ferguson, S. (United Kingdom)
Flaquer, J. (Spain)
Gallo, G. (Italy)
Gavrilova, M. (Canada)
Gudukbay, U. (Turkey)
Gutierrez, D. (Spain)
Havemann, S. (Austria)
Havran, V. (Czech Republic)
Chmielewski, L. (Poland)
Chover, M. (Spain)
Jansen, F. (Netherlands)
Klosowski, J. (United States)
Lee, T. (Taiwan)
Max, N. (United States)
Molla Vaya, R. (Spain)
Murtagh, F. (Ireland)
Pasko, A. (United Kingdom)
Pedrini, H. (Brazil)
Platis, N. (Greece)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Rojas-Sola, J. (Spain)
Rokita, P. (Poland)
Rosenhahn, B. (Germany)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Segura, R. (Spain)
Schumann, H. (Germany)
Skala, V. (Czech Republic)
Slavik, P. (Czech Republic)
Sochor, J. (Czech Republic)
Stroud, I. (Switzerland)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Tokuta, A. (United States)
Vergeest, J. (Netherlands)
Wu, S. (Brazil)
Wuethrich, C. (Germany)
Zara, J. (Czech Republic)
Zemcik, P. (Czech Republic)
Zitova, B. (Czech Republic)

WSCG 2011 was supported by

Sprinx Systems a.s.

SILICON GRAPHICS s.r.o

NVIDIA Corp.

Microsoft, s.r.o. ČR

ZONER Software, a.s.

Faculty of Applied Sciences

Dept. of Computer Science & Engineering

WSCG 2011

Board of Reviewers

Akleman, E. (United States)
Ariu, D. (Italy)
Assarsson, U. (Sweden)
Aveneau, L. (France)
Balcisoy, S. (Turkey)
Battiato, S. (Italy)
Benes, B. (United States)
Benoit, C. (France)
Biasotti, S. (Italy)
Bilbao, J. (Spain)
Biri, V. (France)
Bittner, J. (Czech Republic)
Bosch, C. (France)
Bouatouch, K. (France)
Boukaz, S. (France)
Bouville, C. (France)
Bruni, V. (Italy)
Buehler, K. (Austria)
Cakmak, H. (Germany)
Camahort, E. (Spain)
Capek, M. (Czech Republic)
CarmenJuan-Lizandra, M. (Spain)
Casciola, G. (Italy)
Coquillart, S. (France)
Correa, C. (United States)
Cosker, D. (United Kingdom)
Daniel, M. (France)
de Amicis, r. (Italy)
de Geus, K. (Brazil)
Debelov, V. (Russia)
Domonkos, B. (Hungary)
Drechsler, K. (Germany)
Duke, D. (United Kingdom)
Dupont, F. (France)
Durikovic, R. (Slovakia)
Eisemann, M. (Germany)
Erbacher, R. (United States)
Erleben, K. (Denmark)
Farrugia, J. (France)
Feito, F. (Spain)
Ferguson, S. (United Kingdom)
Fernandes, A. (Portugal)
Flaquer, J. (Spain)
Fontana, M. (Italy)
Fuenfzig, C. (France)
Gallo, G. (Italy)
Galo, M. (Brazil)
Garcia Hernandez, R. (Spain)
Garcia-Alonso, A. (Spain)
Gavrilova, M. (Canada)
Giannini, F. (Italy)
Gonzalez, P. (Spain)
Grau, S. (Spain)
Gudukbay, U. (Turkey)
Guggeri, F. (Italy)
Gutierrez, D. (Spain)
Habel, R. (Austria)
Hall, P. (United Kingdom)
Hansford, D. (United States)
Haro, A. (United States)
Hasler, N. (New Zealand)
Havemann, S. (Austria)
Havran, V. (Czech Republic)
Hernandez, B. (Mexico)
Herout, A. (Czech Republic)
Horain, P. (France)
House, D. (United States)
Chaine, R. (France)

Chaudhuri, D. (India)
Chmielewski, L. (Poland)
Chover, M. (Spain)
Iwasaki, K. (Japan)
Jansen, F. (Netherlands)
Jeschke, S. (Austria)
Jones, M. (United Kingdom)
Jones, M. (United States)
Juettler, B. (Austria)
Kheddar, A. (Japan)
Kim, H. (Korea)
Klosowski, J. (United States)
Kohout, J. (Czech Republic)
Kurillo, G. (United States)
Kyratzi, S. (Greece)
Lanquetin, S. (France)
Lay Herrera, T. (Germany)
Lee, T. (Taiwan)
Lee, S. (Korea)
Leitao, M. (Portugal)
Liu, D. (Taiwan)
Liu, S. (China)
Lutteroth, C. (New Zealand)
Madeiras Pereira, J. (Portugal)
Maierhofer, S. (Austria)
Manzke, M. (Ireland)
Marras, S. (Italy)
Maslov, O. (Russia)
Matey, L. (Spain)
Matkovic, K. (Austria)
Max, N. (United States)
Meng, W. (China)
Mestre, D. (France)
Michoud, B. (France)
Mokhtari, M. (Canada)
Molla Vaya, R. (Spain)
Montrucchio, B. (Italy)
Muehler, K. (Germany)
Murtagh, F. (Ireland)
Nishio, K. (Japan)
OliveiraJunior, P. (Brazil)
Oyarzun Laura, C. (Germany)
Pan, R. (China)
Papaioannou, G. (Greece)
Pasko, A. (United Kingdom)
Pasko, G. (Cyprus)
Patane, G. (Italy)
Patow, G. (Spain)
Pedrini, H. (Brazil)
Peters, J. (United States)
Pina, J. (Spain)
Platis, N. (Greece)
Puig, A. (Spain)
Puppo, E. (Italy)
Purgathofer, W. (Austria)
Reshetov, A. (United States)
Richardson, J. (United States)
Richir, S. (France)
Rojas-Sola, J. (Spain)
Rokita, P. (Poland)
Rosenhahn, B. (Germany)
Rudomin, I. (Mexico)
Sakas, G. (Germany)
Salvetti, O. (Italy)
Sanna, A. (Italy)
Segura, R. (Spain)
Sellent, A. (Germany)
Shesh, A. (United States)
Schultz, T. (United States)
Schumann, H. (Germany)
Sirakov, N. (United States)
Skala, V. (Czech Republic)
Slavik, P. (Czech Republic)
Sochor, J. (Czech Republic)
Sousa, A. (Portugal)
Srubar, S. (Czech Republic)
Stroud, I. (Switzerland)
Subsol, G. (France)
Sundstedt, V. (Sweden)
Tang, M. (China)
Tavares, J. (Portugal)
Teschner, M. (Germany)
Theoharis, T. (Greece)
Theussl, T. (Saudi Arabia)
Tokuta, A. (United States)
Tomori, Z. (Slovakia)
Torrens, F. (Spain)
Trapp, M. (Germany)

WSCG 2011

Communication Papers Proceedings

Contents

Three-dimensional shape descriptors and matching procedures

H. Foulds
School of Computer, Statistical and Mathematical Sciences
North-West University
Potchefstroom, 2531,
South Africa
Henry.Foulds@nwu.ac.za

G.R. Drevin
School of Computer, Statistical and Mathematical Sciences
North-West University
Potchefstroom, 2531,
South Africa
Gunther.Drevin@nwu.ac.za

ABSTRACT

Shape descriptors are used to identify objects in the same way that human fingerprints are used to identify people. Features of an object are extracted by applying functions to the digital representation of the object. These features are structured as a vector which is known as the shape descriptor (feature vector) of that object. The objective when constructing a shape descriptor is to find functions that will yield shape descriptors that can be used to uniquely identify or at least classify an object. A measure of similarity is required to identify or classify an object. The similarity between two objects is computed by applying a distance function to the shape descriptors of the two objects.

The objective of this paper is to examine two of the possible techniques in three-dimensional shape descriptor construction based on Fourier analysis, and to find a descriptor that is able to not only classify, but also identify objects.

Keywords

Shape descriptor, Fourier transform.

1. INTRODUCTION

The Fourier transform has long been used in image and signal processing to convert data from the spatial domain to the frequency domain. Applying the Fourier transform to spatial data results in a set of coefficients that represent different frequency variations in the data. Lower order coefficients represent low frequency variations that normally have large amplitudes while higher order coefficients represent high frequency variations that normally have small amplitudes. The Fourier transform is normally used on data consisting of one dimension (signal processing) or two dimensions (image processing), but can also be applied to three dimensions as seen in Zhang & Chen [Zha01a] and Vranic & Saupe [Vra01a].

Fourier coefficients form conjugate pairs, except for the lowest order coefficient. To create a descriptor from Fourier coefficients the magnitudes of the coefficients are calculated. The first K of these values, corresponding to the K lowest Fourier coefficients, are used. K is a threshold value to establish the smallest number of coefficients needed to identify each object.

In this paper two methods are developed that can be used with the Fourier transform to identify objects. In both methods a $\theta\varphi$-matrix is created to which a Fourier transform is applied to create feature vectors.

The data used in this project are in the form of three-dimensional triangle mesh models. In a triangle mesh model each object is approximated by a collection of structured triangles. The triangles represent the faces of the mesh model and each face has three vertices. Some of the faces share common vertices or sides. The faces are represented by an $N\times3$ matrix with N the number of faces. The vertices are represented by an $M\times3$ matrix with M the number of unique vertices. The three values in each row in the faces matrix represent the three vertices of a face. Each value is the row number of a vertex in the vertices matrix. The first step in calculating the

descriptor of an object is to obtain the N×3 matrix representing the N faces and the M×3 matrix representing the M vertices that approximate the object.

Matching objects when they do not have the same pose causes a serious problem. To solve this problem the descriptor has to be translation and rotation invariant. The most frequent used method to solve this problem is to apply Principle Component Analysis (PCA, Hotelling transform, Karhunen-Loeve transform) to the triangle mesh of an object before the descriptor is calculated [Zha01a, Vra01a, Vra03a, Pap06a]. The result of doing PCA is that all objects that are similar will have similar orientations and the descriptors that are calculated will have a smaller match distance. PCA can be used as a second step in calculating a descriptor to ensure rotation and translation invariance when the descriptor technique itself is not invariant to rotation and translation. An example of PCA can be seen in Figure 1. When classifying objects scale invariance is required. The objective of this project was to solve an exact matching problem, therefore scale variance was not considered.

The PCA method uses the eigenvalues and eigenvectors, calculated from the covariance matrix of the vertices matrix, to rotate an object in such a way that the first principal axis aligns with the x-axis, the second principal axis with the y-axis and the third principal axis with the z-axis. On completion of PCA the largest variance of mass is in the x-axis direction. For highly symmetrical objects the eigenvalues are very similar. The small differences in symmetrical objects cause errors during PCA because the eigenvalues are similar and a small change in the eigenvalues causes the eigenvectors to change dramatically. Figure 2 shows the result of applying PCA on symmetrical objects. Using PCA to normalize pose when objects are symmetrical is not very successful.

2. REPRESENTATION

Centroid distance is defined as the distance from the centroid of an object to the surface of that object. The object is centred and orientated using PCA. After PCA the centroid distance is calculated for angles θ and φ with $\theta \in [0,\pi]$ and $\varphi \in [0,2\pi]$. θ is the polar angle from the z-axis and φ the azimuthal angle in the xy-plane from the x-axis. The centroid of the object is located at the origin of the coordinate framework. A two-dimensional θφ-matrix is created where the rows represent θ-values and the columns represent φ-values. The value located at index [θ,φ] of the matrix is the distance from the centroid, in the direction (θ,φ), to the surface of the object. If the surface of the object is intersected more than once, the maximum value is used. In the result for $\theta \in [0,2\pi]$ the elements in the matrix for $\theta \in [\pi,2\pi]$ will be a mirror of the values for the elements with $\theta \in [0,\pi]$. For this reason $\theta \in [0,\pi]$ is used.

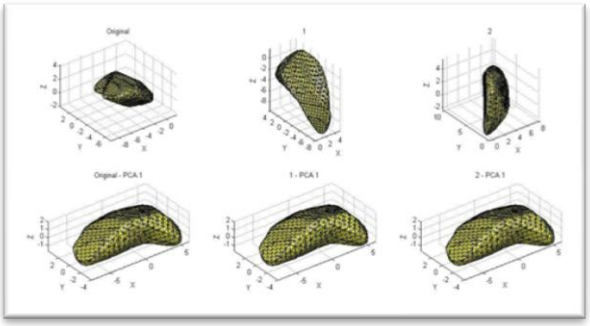


Figure 1. The top row shows three different orientations of the same object. The bottom row shows the result of applying PCA to the top row.

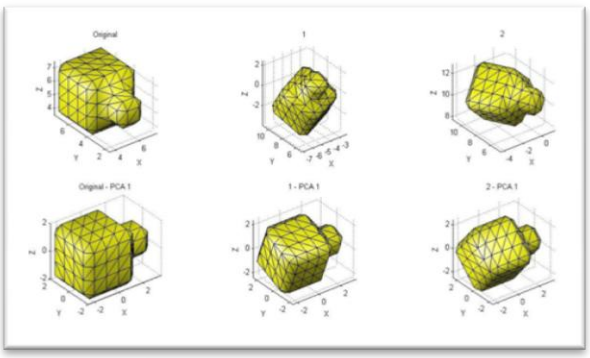


Figure 2. The top row shows three different orientations of a symmetrical object. The bottom row shows the result of applying PCA to the top row.

The discrete Fourier transform (DFT) of the θφ-matrix is calculated and the result used to create the feature vector. The feature vector is created by applying a method, similar to the process used by Vranic & Saupe [Vra01a], to the Fourier coefficients. For a chosen integer value K, the (2K+1)×(2K+1) matrix centred on the lowest frequency coefficient is returned. Except for the lowest frequency coefficient, all other coefficients form conjugate pairs. The feature vector is defined as the magnitudes of the Fourier coefficients. The coefficients are ordered according to distance from the lowest frequency coefficient.

Algorithm:

1. Get faces matrix and vertices matrix
2. Do PCA
3. Centroid is located at origin after PCA.

4. Get θφ-matrix by calculating distance from centroid to last triangle intersected for each [θ,φ] (Two different methods are discussed in paragraph 2.1 and 2.2)
5. Calculate 2D Fourier transform of θφ-matrix
6. Set the number of Fourier coefficients by choosing value for K
7. Get the feature matrix by calculating the absolute value of the elements in the (2K+1)×(2K+1) matrix in the centre of the Fourier transformed θφ-matrix (centred on the lowest frequency coefficient)
8. Calculate the 2D Euclidean distance of each element in the feature matrix from the centre
9. Reorder the elements in the feature matrix into a one-dimensional array, sorting them according to the distance calculated in step 8 (For elements at the same distance, the same order is used for each object as they are orientated with PCA)
10. The result is the feature vector.

Two techniques to calculate the distance from the centroid to the surface of an object were evaluated.

2.1. Method 1

In this technique four θφ-matrices are created with $\theta \in [0,\pi]$ and $\varphi \in [0,2\pi]$. Increments of 1, 4, 9 and 18 degrees for θ and φ are used for each of the four matrices respectively. Larger increments will result in smaller matrices. For each of the directions (θ,φ) in the θφ-matrix the distance is calculated from the centroid to an intersection with a face. If more than one face is intersected, the maximum distance is used. Each face may be intersected multiple times for different directions (θ,φ).

2.2. Method 2

In the second technique four θφ-matrices are also created with $\theta \in [0,\pi]$ and $\varphi \in [0,2\pi]$. Increments of 1, 4, 9 and 18 degrees for θ and φ are used for each of the four matrices respectively. For each of the faces a range of θ and φ values is determined to create a region surrounding the face. Depending on the increment length, the number of intervals for θ and φ are calculated.

This number of intervals is used to create a set of points on the face defined by intervals of λ and β in $\lambda E + \beta F + (1 - \lambda - \beta) G$ with $0 \le \lambda \le 1$, $0 \le \beta \le 1$ and $0 \le \lambda+\beta \le 1$. E, F and G are the vertices of the face. The distance from the centroid to the plane defined by the vertices of the face is calculated for the directions (θ,φ) for all the points in the set. This procedure ensures that distances are calculated only for the set of points on a face.

Using method 2 speeds up the process of calculating the θφ-matrix considerably as there is no need to repeat the calculations for all the directions of all the faces. The process is applied to each face and the results obtained for all the faces are combined. The results obtained with these two techniques are given in section 4.1.

3. MATCHING

3.1. Distance functions

Distance functions are used to determine the distance between two feature vectors f_1 and f_2. A number of distance functions are defined, among others, by Vranić [Vra03a], Osada *et al*. [Osa01a] and Long *et al.* [Lon02a]. Distance functions used in this project are:

- l_1 norm

$$d_1(\mathbf{f}_1, \mathbf{f}_2) = \|\mathbf{f}_1 - \mathbf{f}_2\|_1 = \sum_{i=1}^{N} |f_{1i} - f_{2i}| \qquad (1)$$

This function defines the distance between two feature vectors as the sum of the absolute values of the differences between each set of corresponding elements.

- l_2 norm

$$d_2(\mathbf{f}_1, \mathbf{f}_2) = \|\mathbf{f}_1 - \mathbf{f}_2\|_2 = \sqrt{\sum_{i=1}^{N} (f_{1i} - f_{2i})^2} \qquad (2)$$

With this function the distance between two feature vectors is the square root of the sum of the squares of the differences between each set of corresponding elements.

- Minkowski with p=0.8

The next two distance measures use the Minkowski norm given in Long *et al*. [Lon02a] as

$$d(\mathbf{f}_1, \mathbf{f}_2) = \left(\sum_{i=1}^{N} |f_{1i} - f_{2i}|^p \right)^{\frac{1}{p}}. \qquad (3)$$

The value of p was chosen as 0.8 for this norm.

- Minkowski with p=1.2

This norm uses the same form as in Equation (3), but with p chosen as 1.2.

- l_{max} norm

The l_{max} norm is the maximum of the absolute values of the differences between the sets of corresponding elements.

$$d_\infty(\mathbf{f}_1, \mathbf{f}_2) = \|\mathbf{f}_1 - \mathbf{f}_2\|_\infty = \max_{1 \le i \le N} |f_{1i} - f_{2i}| \qquad (4)$$

3.2. The matching process

Matching is done by first applying the same descriptor method to two different objects after which the distance between the two feature vectors is calculated. This distance is used as a measure of the similarity of the two objects.

When working with sets of objects, a matrix of distance values is created to match two sets of objects. The rows represent the objects of the first set and the columns represent the objects of the second set. Each value in the matrix is used as a measure of the similarity between two objects, an object from the first set as defined by the row number and an object from the second set as defined by the column number. For each object in the first set, the closest match is found from the second set. With this technique an object can be identified from a set of possible objects.

The elements in a similarity matrix M are calculated using one of the distance functions, therefore $M(i,j)=d(f_i,f_j)$ where f_i is the feature vector of object i on row i of matrix M and f_j is the feature vector of object j on row j of matrix M. d is the chosen distance function.

In this study three sets of 60 objects are used to test the usability of the descriptors for identification. Examples of these objects are given in Figure 3. Each of the three datasets used in the matching process consist of 600 triangle mesh models representing the 60 objects. Each object is digitized 10 times resulting in 10 representations of each object. Due to the nature of the digitization process there are small variations in the representations of the objects. The 600 models in each dataset are divided into two sets with five models representing each object in a set. During the matching process the 300 models representing 60 objects from the first set are matched to the 300 models representing the same 60 objects from the second set. The matching process creates a similarity matrix with the models from the first set represented by 300 rows and the models from the second set represented by 300 columns. Each element in the matrix is the result of a distance function applied to the object represented by the row from the first set and the object represented by the column from the second set. Five different distance functions are used to create five different similarity matrices. Because multiple rows and columns are used to represent feature vectors calculated from different representations of the same object, aggregates of these columns and rows are taken. The aggregate functions used are minimum, mean, maximum and sum. This results in a matrix with 60 rows and 60 columns. The objective is to identify and match objects in a set using similar representations of the same objects in another set. Because both sets contain models of all 60 objects, an object in a specific row must be matched with an object in a specific column. This is done by finding the object with the smallest distance function result.

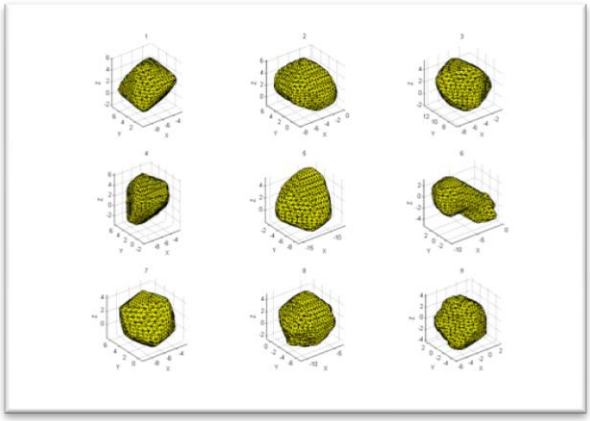


Figure 3. Representations of the first nine objects in dataset 1 used during matching.

4. RESULTS

4.1. Feature vector creation

The project was implemented using Matlab 2007 on an Intel Core2 Quad 2.4GHz PC. The first tests are to show the results obtained when applying the two techniques to calculate the θφ-matrix as discussed in section 2.1. A sphere with radius 3 consisting of 88 faces is used to evaluate the two techniques.

The results when calculating the θφ-matrix of the sphere using method 1 are shown in Table 1. The first column lists the sizes of increments used for θ and φ and the second column lists the time it took to create the θφ-matrix using method 1. The third column show the time it took to create the θφ-matrix using method 2.

Increments (degrees)	Time (s) Method 1	Time (s) Method 2
1	630	506
4	40	0.9
9	9	0.2
18	2.3	0.06

Table 1. Time needed to calculate θφ-matrix using method 1 and 2.

Calculating the θφ-matrix is considerably faster when using method 2 and incrementing θ and φ with 4 degrees or more as seen in Table 1.

Figure 4 shows the θφ-matrix for increments of 1 degree obtained using method 2. The next step is to test the technique used for the creation of the Fourier descriptors. Figure 5 shows 3 representations of objects used to construct the feature vectors. Object A is a sphere consisting of 88 faces. Object B is a rounded cube consisting of 188 faces. Objects C and D are irregular objects consisting of 2476 and 1372 faces respectively. The results of constructing the feature vectors are listed in Table 2. The θφ-matrix is calculated using the second technique with increments of 4 degrees. The feature vectors are calculated with threshold K=10. Figure 6, 7 and 8 shows the θφ-matrix, Fourier coefficients and feature vector for object C.

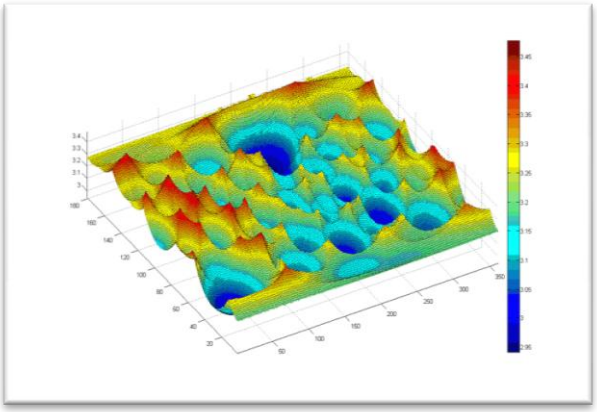


Figure 4. θφ-matrix using method 2 with increments of 1 degree.

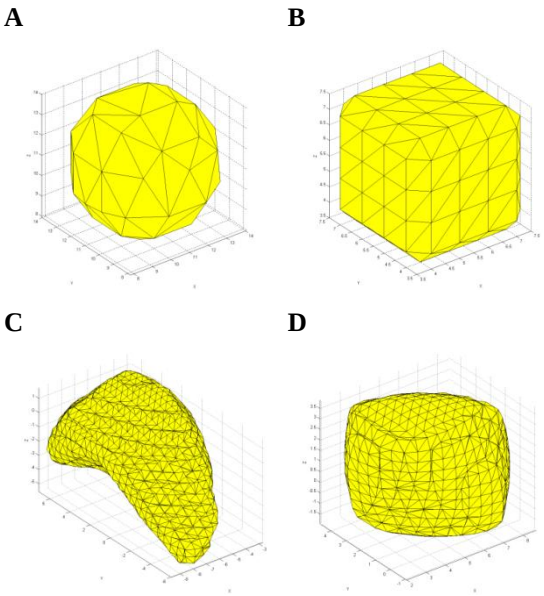


Figure 5. (A) A sphere with radius 3 consisting of 88 faces. (B) A rounded cube consisting of 188 faces. (C) An object consisting 2476 faces. (D) An object consisting of 1372 faces

Object	Time (s) Method 2
A	0.7
B	0.9
C	4.4
D	3.5

Table 2. θφ-matrix obtained by using method 2 with increments of 4 degrees.

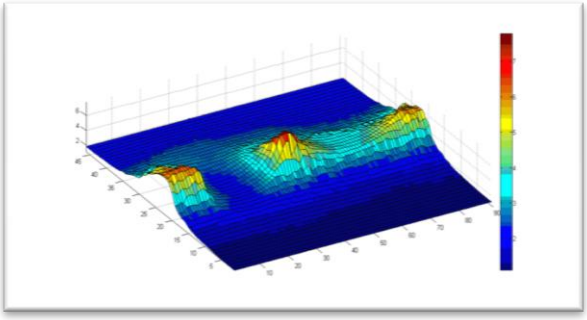


Figure 6. θφ-matrix for object C using method 2 with increments of 4 degrees.

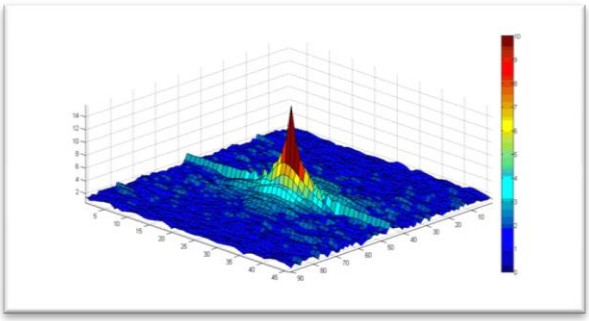


Figure 7. Fourier coefficients obtained from the DFT of the θφ-matrix for object C.

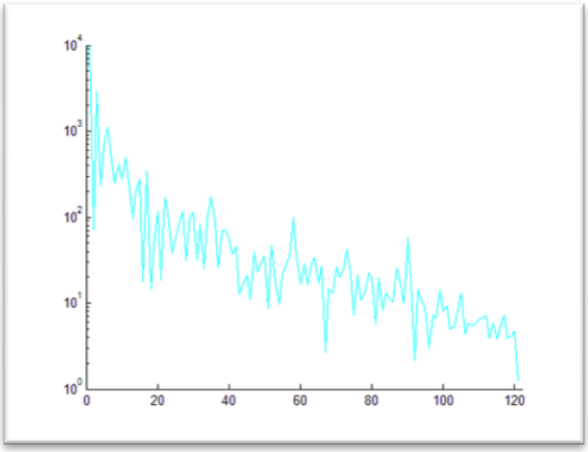


Figure 8. Feature vector constructed from Fourier coefficients for object C

Figure 9 shows the feature vectors of 9 objects. Object A is the sphere (object A) consisting of 88 faces in Figure 5. Object B is a sphere consisting of 66 faces. Object C is a rounded cube consisting of 188 faces. Objects D through G are different representations of object C in Figure 5, consisting of 2384, 2268, 2476 and 2444 faces. Object H and I are very similar objects consisting of 1372 and 1932 faces respectively. Different colours are used to indicate the different objects. Below is a list of colours used.

- Object A - (green).
- Object B - (dark green).
- Object C - (red).
- Object D - (blue).
- Object E - (dark blue).
- Object F - (cyan).
- Object G - (dark cyan).
- Object H - (purple).
- Object I - (magenta).

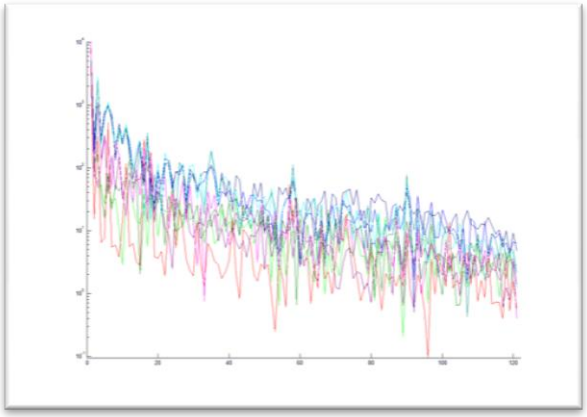


Figure 9. Feature vectors of 9 objects, also represented in the Figure 10.

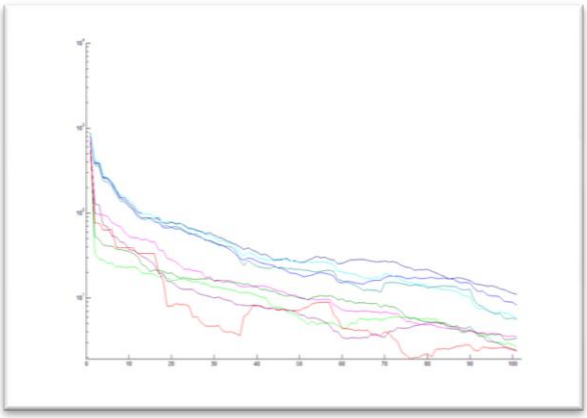


Figure 10. Moving average of length 21 of the feature vectors in Figure 9.

Smoothing of feature vectors as in Figure 10 is done only to improve the visualization of the feature vectors for printing. The original feature vectors are used in the matching process as smoothing causes a loss of feature information. Smoothing is done using a moving average of length 21.

When comparing the feature vectors in Figure 10 it is clear that the four feature vectors at the top are grouped together. These feature vectors are for objects D, E, F and G. When comparing the four triangle mesh models, it is clear that they are different representations of the same object, hence the similarity in their feature vectors. Objects A, B, H and I are very rounded in shape, and their feature vectors are also very close together.

For method 2 the θφ-matrix is calculated with increments of 4, 6 and 10 degrees for angles θ and φ giving 4050, 1800 and 648 elements for each of the matrices. Using increments of 1 degree result in excessive processing time requirements. Using large increment sizes produce feature vectors that result in inaccurate matching. For this reason increments of 4, 6 and 10 degrees where chosen.

The time needed to calculate the θφ-matrices for the different increments are compared in Figure 11. In Figure 11 the number of faces is not the only influence on processing time. The rightmost four objects in the figure are very similar, and variations in the complexity and orientation of the faces influences the processing time. Increasing the increment sizes for angles θ and φ decreases calculation speed, but causes a loss of accuracy.

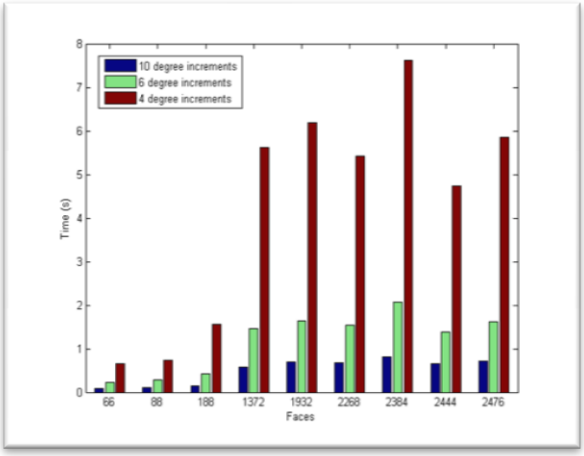


Figure 11. Processing time required to calculate the θφ-matrix for the 9 objects of Figure 9 using method 2 with increments of 10, 6 and 4 degrees.

4.2. Matching

The feature vectors are calculated with K=5 resulting in 36 elements in each feature vector. These feature vectors are calculated using increments of 10 degrees for angles θ and φ. It takes 350 seconds to calculate the feature vectors of all 600 mesh models in a dataset using increments of 10 degrees. For increments of 6 degrees it takes 940 seconds, and 2650 seconds for increments of 4 degrees. The process of calculating the θφ-matrix takes up most of the time during feature vector generation. Four aggregate functions are applied to the results of each of the distance functions, as discussed in section 3.2. The results for each of these functions are listed in the columns marked "Min", "Mean", "Max" and "Sum". The results are the number of errors made in matching the two sets of mesh models. Figure 12 contains four similarity matrices for the l_1 norm distance function calculated from the results of the four aggregate functions. Colours closer to blue indicate similar objects, while colours closer to red indicate dissimilar objects.

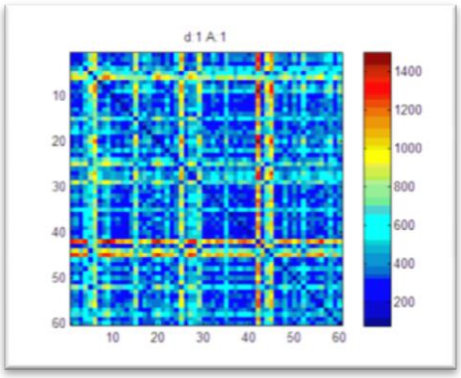


Figure 12. A similarity matrix for the l_1 norm distance function obtained from Fourier method 1 using Min aggregate. Colours closer to blue indicate similar objects, while colours closer to red indicate dissimilar objects.

The first elements of the feature vectors correspond to the lower order Fourier coefficients. The lower order Fourier coefficients relate to large changes in shape while the higher order coefficients relate to smaller variations in shape. For this reason only a number of coefficients surrounding the first coefficient is needed to create a feature vector. This approach will also reduce the influence of noise, which is usually associated with small variations. As the first elements have very large values, they will have a greater influence during the distance calculations between feature vectors. Therefore large variations in shape will result in bigger differences between feature vectors. From the results it is clear that this feature vector creation method will produce feature vectors that can be used to identify objects.

The results of the matching process are given in Tables 3 to 5. These results show that the four distance measures performed well during the matching process. Table 5 shows that without PCA the results are inferior. The errors made during identification were of objects that are exceptionally similar.

A good method for identification should have a low probability of false acceptance and a low probability of false rejection. The area closer to the origin in the ROC graph will reveal the more accurate method. For this reason the ROC graphs are displayed for a probability of false acceptance and false rejection up to 50%. The ROC graphs in Figure 13 illustrate that the distance functions yield very similar results. The two graphs in Figure 13 yielding poor results are the Kullback-Leibler divergence and Jeffrey divergence.

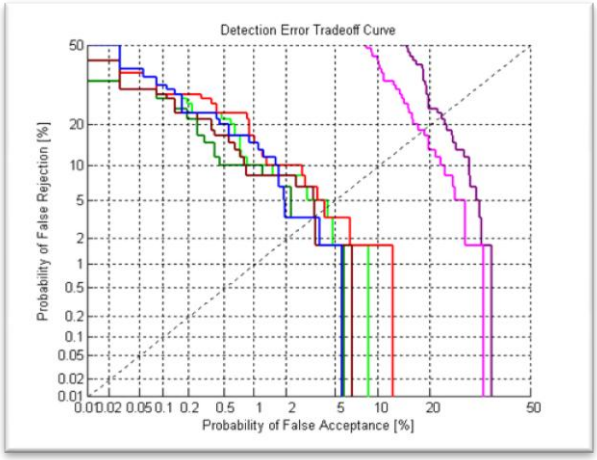


Figure 13. ROC graph of matching results for l_1 norm distance function applied to feature vectors generated using method 2.

These two distance functions gave poor results in all the preliminary tests and where excluded from the rest of the study. Even though using smaller increments result in better matches, the differences between the results observed when incrementing angles θ and φ by 10, 6 and 4 degrees are very small. The time needed for processing make increments smaller than 10 ineffective. With all techniques the "minimum" aggregate gave better results and "maximum" aggregate gave inferior results.

Distance function	Min	Mean	Max	Sum
$l_1$norm	1	2	22	2
l_2 norm	3	10	31	10
Minkowski with $p = 0.8$	1	2	18	2
Minkowski with $p = 1.2$	2	2	24	2
$l_{\max}$ norm	4	18	36	18

Table 3. Results for dataset 3 created with method 2 and increments of 10 degrees.

Distance function	Min	Mean	Max	Sum
$l_1$norm	1	3	12	3
l_2 norm	0	7	23	7
Minkowski with $p = 0.8$	2	3	13	3
Minkowski with $p = 1.2$	0	4	16	4
$l_{\max}$ norm	5	11	27	11

Table 4. Results for dataset 3 created with method 2 and increments of 4 degrees.

Distance function	Min	Mean	Max	Sum
$l_1$norm	30	49	52	49
l_2 norm	31	52	55	52
Minkowski with $p = 0.8$	30	47	51	47
Minkowski with $p = 1.2$	29	49	52	49
$l_{\max}$ norm	39	50	53	50

Table 5. Results for dataset 1 created with method 2 and increments of 10 degrees. No PCA is applied.

5. CONCLUSION

Principal Component Analysis (PCA) plays a vital part in normalizing the orientation of objects during identification. Symmetrical objects result in errors during PCA. Two methods to obtain feature vectors, using the Fourier transform, are described in this paper. Fourier methods are effective in identifying objects as they are fast and accurate. Their only drawback is that accuracy decreases when large numbers of objects in the datasets are very similar. Of the five distance measures evaluated the l_1 norm, l_2 norm, Minkowski with p=0.8 and Minkowski with p=1.2 distance measures are best suited for identification.

6. FUTURE WORK

Alternative methods to normalize orientation can also be explored to improve results.

The CSS descriptor method [Dre05a, Zha01b] is another technique that could possibly be adapted for three-dimensional identification in future research.

7. ACKNOWLEDGMENTS

The authors would also like to thank Keith Forbes and Colin Andrew for their aid during this project.

8. REFERENCES

[Dre05a] Drew, M.S., Lee, T.K. and Rova, A. Shape Retrieval with Eigen-CSS Search. Technical Report, TR 2005-07, Simon Fraser University, Vancouver, B.C., Canada, School of Computing Science, Vancouver, B.C., Canada. 2005.

[Lon02a] Long, F., Zhang, H., and Feng, D.D. Fundamentals Of Content-based Image retrieval. In D. Feng, W. Siu, & H. Zhang, Multimedia Information Retrieval and Management - Technological Fundamentals And Applications (pp. 1-26). Springer, 2002.

[Osa01a] Osada, R., Funkhouser, T., Chazelle B. and Dobkin, D. Matching 3D Models with Shape Distributions. Proceedings of the International Conference on Shape Modeling & Applications, 2001.

[Pap06a] Papadakis, P., Pratikakis, I., Perantonis, S. and Theoharis, T. A Concrete Radialized Spherical Projection Descriptor for 3d Shape Retrieval. IEEE International Conference on Shape Modeling and Applications (SMI'06), 2006.

[Vra01a] Vranić, D.V. and Saupe, D. 3D Shape Descriptor Based on 3D Fourier Transform. Proceedings of the EURASIP Conference on Digital Signal Processing for Multimedia Communications and Services (ECMCS 2001), 2001.

[Vra03a] Vranić, D.V. 3D Model Retrieval. PhD Dissertation, Universität Leipzig, Institut für Informatik, 2003.

[Zha01a] Zhang, C. and Chen, T. Efficient Feature Extraction for 2d/3d Objects in Mesh Representation. International Conference on Image Processing, 2001.

[Zha01b] Zhang, S. and Lu, G. Content-Based Shape Retrieval Using Different Shape Descriptors: A Comparative Study. IEEE ICME. Tokyo, Japan. 2001.

Comparative Performance of GPU, SIMD and OpenMP Systems for Raw Template Matching in Computer Vision

Juan Mendez
Departamento de Informática y Sistemas. Universidad de Las Palmas de Gran Canaria
35017, Las Palmas, Spain
jmendez@dis.ulpgc.es

Javier Lorenzo
SIANI. Universidad de Las Palmas de Gran Canaria.
35017, Las Palmas, Spain
jlorenzo@iusiani.ulpgc.es

Modesto Castrillon
SIANI, Universidad de Las Palmas de Gran Canaria.
35017, Las Palmas, Spain
mcastrillon@iusiani.ulpgc.es

ABSTRACT

Template matching is a traditional technique of Computer Vision whose advantages and disadvantages are known. However, advances in computer hardware allow computing it effectively with the use of SIMD instruction set, GPUs or multi-core systems. The computation of that low-level primitive in sub millisecond scale would improve high theoretical methods if they are used with high efficient primitives. This paper presents the comparative results of basic template matching by using SIMD instructions, multi-core systems and multi-GPU implementations. The results of this study will show that the high-specialized instruction in modern releases of SIMD and the use of multi-core systems outperforms the implementations based on GPUs for small mask size due to memory transfer cost. However, for big mask size GPU and SIMD systems have similar performance.

Keywords

Computer Vision, Template Matching, Parallel Computing, GPU, Multi-Core Systems.

1. INTRODUCTION

Template Matching is a Computer Vision procedure focused on the detection of local features in a image that seems or resembles similar properties than a small part of the image or mask. This is a general definition and many different approaches implement the concept by using different paradigms. The main drawback of template matching is that it implies a "wasteful" exploring of the image for searching a local area very similar to the mask. This requires the *sliding* of the mask across the entire picture and the computation of some measure of similarity or distance for each position.

Computer Vision applications have two opposite constraints. The first is related to the randomness of the data that requires the use of higher-level theoretical procedures. These methods allow robust procedures that deal efficiently with the enormous variability of the data and provide stable results. The second constraint is related to the computational efficiency that tends to carry out real-time performance to fit the application needs and can be useful in real world problems. The progresses in Computer Vision deal to advances in both directions. The equilibrium between both viewpoints determines the most useful procedures for a defined state of the advances in the computer technology and in theoretical methods.

The evolution of computer hardware can revalue some traditional and simple procedures because they will be computed faster than in older implementations. These fast and simple procedures can be used as basic results, which are the first level of intermediate results, in more elaborated and complex procedures. That is, they can be considered primitives rather than complete procedures. Raw Template Matching is a traditional Computer Vision technique with advantages and drawbacks, but it can be computed very efficiently in modern computer hardware as Graphics Processor Unit (GPU) and Single Instruction Multiple Data (SIMD) arithmetic units in multi-Core systems.

Modern computer ranging from desktop to rack servers have multiple cores as well as some GPU in standard hardware configuration. Also many processors, such as the Intel/AMD series, include vector units that allow advanced SIMD instructions. Thus, no additional cost is needed in order to have high performance computer hardware. However, these units, that are normally unused, are difficult to integrate in standard programming code unless no special programming libraries are used. That is a very common tendency in many branches of computing where hardware advances are faster than programming techniques, or programming practices. The introduction of special libraries such as OpenCV [Bra08a] provides to the user the advantage of high computer performance and hides its complexity. But OpenCV library is a bit conservative and does not take advantage of all the features that the hardware can provide. The best performance using SIMD is achieved when the assembler code is used. Developing a complete application in assembler level is not a good idea, but coding only small pieces of high efficient primitives can be a good option in some special cases. Although programming GPU is difficult, the use of this specialized hardware is useful only if their programming is hidden in specialized libraries for small but high efficient pieces of the software used for Computer Vision.

The main idea of advanced tracking systems such as Lukas-Kanade procedure [Luk81a] is based on the assumption that small changes in image motion such as the brightness constancy and spatial coherence of small areas in the image motion can be detected; that is, for short time intervals, small masks can be used to detect the motion of real world objects. The similarity between mask and local image area can be computed by using different features and classification strategies and methods, but the one based on the similarity or distance between the raw data at pixel level is the simplest of all.

According to Brunelli [Bru09a], the main drawback of Template Matching is its high computational cost, which has two distinct sources. The first one is the necessity of using multiple templates to capture the variability exhibited by the appearance of complex objects. The second one is related to the size of templates: the higher the resolution, the heavier the computational requirements. Raw template matching is much more simple if compared with advanced matching that incorporates geometric invariance [Ull04a] [Kin07a], of feature characterization rotation invariant of mask based on moments of Hu and Zernike [Teh88a]. In the case of advances tracking applications, template matching must be used along with higher level procedures such as Kalman and Particle filter [For02a]. Template Matching is also a basic tool used in video encoding, where the correspondence points between successive frames in video image, eg. in MPEG video compression, implies the detection of image block of 16x16 pixels in previous frames [Sha01a]. However, in video compression the required matching is carried out in a narrow area bounding the block.

This paper presents a comparative study of implementation of raw template matching based on modern technologies using GPU and SIMD architectures that allows the computation of template matching of small masks in few milliseconds. The basic results of raw template matching are presented as well as its efficient implementation in the more modern release of SIMD instruction set. The details of the implementation in GPU are also presented, and finally the comparative results of both approaches by using multiple cores with OpenMP [Cha08a] parallel programming.

2. RAW TEMPLATE MATCHING

The easiest way to achieve raw template matching is by using a similarity or distance measure between a local area of the image and a mask or template. If the matching is based on a distance measure, it is necessary to find the minimum or minima. A widespread used distance measure in different branches of Mathematics and Computer Science is the one based on a vector norm, e.g. as the based on the Minkowski metric [Bru09a] [Har01a]. The matching result $\boldsymbol{R}$ can be computed from the image data $\boldsymbol{D}$ and the template or mask $\boldsymbol{M}$ as:

$$\boldsymbol{R}_p = \left[\| \boldsymbol{D}(x+u, y+v) - \boldsymbol{M}(u,v) \|_p \right]^p$$

where $\| A \|_p$ is the L_p norm. Examples of very used norms in Mathematics and Computer Vision are: L_1, L_2 and L_∞. The definition of template matching using multichannel images and masks in these cases is the following:

$$R_1(x,y) = \sum_c \sum_{u,v} |D_c(x+u, y+v) - M_c(u,v)|$$

$$R_2(x,y) = \sum_c \sum_{u,v} |D_c(x+u, y+v) - M_c(u,v)|^2$$

$$R_\infty(x,y) = \max_{c,u,v} |D_c(x+u, y+v) - M_c(u,v)|$$

The L_2, norm which generates the template matching R_2, is used by OpenCV Library [Bra08a], although it has not the lowest computational cost. OpenCV is originally based on performance primitives of Intel/AMD processors, but these systems are better suited to compute efficiently the L_1 norm, which is based on the computation of Sum of Absolute

Differences (SAD). Also NVIDIA GPUs have basic support for computing the SAD primitive. The SAD for two arrays is defined as: $SAD(\boldsymbol{A},\boldsymbol{B}) = \sum_i |A_i - B_i|$. Although the use of L_1 is no advantageous in general purpose programming, some special hardware makes it the best choice. However, nowadays the special SIMD hardware of Intel processors is of such a common and widely use that we can call it as general purpose hardware.

Instead of using a coordinate system placed on the center of the template mask, we will use upper-left corner centered coordinates. It is more advantageous to deal with memory alignment, because it plays a main role in efficient memory accesses. The real position of the detection of the mask can be obtained after the minimum detection by using a simple offset to the mask center. We will use the offset evaluation of the match for a mask of dimension $S_x \times S_y$defined as:

$$R(x,y) = \sum_{u=0}^{S_x-1}\sum_{v=0}^{S_y-1} |D(x+u, y+v) - M(u,v)|$$

Multi-channel template matching, e.g. in RGB images, can be obtained simply by adding the results obtained in the previous equation that was applied in every image channel and mask. For a $N_x \times N_y$ image, the border band, which is usually located bounding the image, is moved to the right and low of the image. The right null band is $S_x - 1$ width and low band is $S_y - 1$ high. In the previous Equation the (x, y) values run in the intervals: $x \in [0, N_x - S_x + 1]$ and $y \in [0, N_y - S_y + 1]$. Usually, the border band is set to null value, but to avoid any problem with the minimum computation we decide to set it to the highest numeric positive value in its internal representation. After the local detection of the minimum matching value, its location must be offset by $(S\_x/2, S\_y/2)$. Template matching in 2D has advantages related to the data alignment, and it also can be computed by row 1D oriented matching:

$$R(x,y) = \sum_{u=0}^{S_x-1}\left[\sum_{v=0}^{S_y-1} |D_{y+v}(x+u) - M_v(u)|\right]$$

That can be computed by using the following general 1D template matching:

$$B(x) = \sum_{u=0}^{S-1} |A(x+u) - C(u)|$$

The use of Region of Interest (ROI) reduces significantly the computational cost because the template search can be reduced to a fraction of the image area. The ROI usage for tracking requires the implementation of a strategy for updating the ROI according to the detected trajectory by using a predictive filter such as the one based on Kalman or Particle Filters. In this paper we have computed the worst case, when the maximal ROI extends to the entire image. This option allows us to obtain an upper bound of the tracking computational time.

The two main problems involved in the computation of template matching in 2D and 1D are arithmetic and memory access. The arithmetic is concerning to the computation of SAD, which is not cheap if no special hardware is available. Memory access is a less evident problem, but it is more important in modern computers because their performance is mainly related to the pattern of memory access. The sliding of the mask across the entire image requires that all the different memory alignments patterns must be used. The sliding u value between $B(x)$ and $A(x+u)$ is the cause of many of the low performance issues in computer applications because it is very important in memory access efficiency.

3. SIMD-BASED TEMPLATE MATCHING

The efficient implementation of template matching in modern computer architectures requires a revision of some specialized instructions of the machine code of some popular microprocessors. Unfortunately, those instructions are not used by the compilers to translate the user code written in high level languages as C/C++ to machine code. This implies that the user must code the parts of the software dealing with the specialized instructions using assembly language or inline embedded assembly. In this section only some guidelines of the specialized instructions are shown with the aim of being useful for researchers and developers in Computer Vision.

Early versions of SIMD in Intel/AMD processors incorporate a SAD instruction called *psadbw* in the first SSE (Streaming SIMD Extension) release of the MMX (Multi Media eXtension) instruction set. It uses the MMX registers of 64 bits, allowing computing the SAD for 8 bits unsigned integer data. This instruction allows improving the arithmetic part of the array matching but does not solve the problems related to the sliding of the mask array across the data array. Multiple unaligned data read were needed to perform and achieve the whole matching.

A recent SSE extension, that use 128 bit registers, has included the *mpsadbw* instruction [Int09a] that solves this problem by including in-hardware sliding computation, which avoids that the user must design a code with unaligned data load. This improvement introduced in SIMD release SSE4.1 allows higher

performance, but at the time of this study, it is not included yet in all computer vision libraries, e.g. OpenCV. This instruction computes multiple and sliding SAD for 4 bytes masks including an immediate additional argument (*imm8*) that controls the selection of the group of 4 bytes defining the mask and also controlling the sliding option. The *mpsadbw* instruction requires three arguments: the source register containing 16 byte mask data, s(0-15), the result register that initially contains 16 byte data from an image row, d(0-15), and finally one byte register, imm8, to control the instruction mode. The result is obtained as 8 unsigned short array r(0-7).

This instruction is extremely important for modern HDTV codecs, and allows an 8x8 block difference to be computed in fewer than seven cycles [Kua07a], but also is very useful in general template matching of small mask on the whole image. This instruction implements the computation of the SAD for 4 unsigned char integer values and the in-hardware computation of the sliding of the 4 bytes across the data register. If $imm8(2) = a \in \{0,1\}$ and $imm8(0-1) = b \in \{0,1,2,3\}$, it computes for $i = 0, \ldots, 7$

$$r(4a+i) = \sum_{j=0}^{3} |d(4a+i+j) - s(4b+j)|$$

Computing the whole sliding of a mask across a data array will require a more complex arrangement. Therefore, to compute a full matching of a data array with a mask of suitable size multiple of 4 bytes, we have designed a computational arrangement, which is shown in Figure 1, as an useful chart that allows an easy implementation for Computer Vision developers and researchers. In this chart, Data and Result are the arrays involved in a general 1D matching. The first array is 8 bits unsigned integer and the second array 16 bits unsigned integers, also it is used the 8 bit unsigned int Mask array, whose length is multiple of 4. In Intel architecture the SSE instructions for loading and storing data are penalized if the memory reference is not aligned to 16 bytes. The goal of the arrangement shown in the Figure 1 is the computation of Result(0-7) and Result(8-15) for different mask sizes. Data are read from memory in 16 bytes block such as the CPU can read Data(0-15) and Data(16-31). However, it can be read Data(8-23) in unaligned way by incurring in efficiency penalties. To avoid that, it can be obtained Data(8-23) by using register instructions from the aligned Data(0-15) and Data(16-31), which can be read without penalties. Data contained in the first row are directly read and the contained in the second row are obtained from the previous by using register operations.

Column containing Result(0-7) defines the intermediate results that must be added to get the result and from which they are obtained. For instance, to compute Result(0-7) by using the smaller mask of 4, we must compute by using the imm8 value of 000 in the Data(0-15) according to the description of the instruction. When we want the same result but for a mask size of 8, we must obtain the previous result (using 000 in Data(0-15)) and add this to the intermediate result by using 010 applied to Data(8-23). For each row related to a mask size, intermediate results are organized form right to left and successively computed from Data(0-15), Data(8-23), Data(16-31), Data(24-39) and Data(32-47). Each cell in the sub-rows corresponds to each mask size when it is computed by using the defined {imm8} datum. Although the arrangement can be extended to bigger masks of size $4 \times N$, we only have included the cases from 4 to 32.

4. TEMPLATE MATCHING IN GPU

A GPU has many processors or cores that can be suitably arranged to fit better for a specific problem. The advantage of GPUs is the massive number of cores, e.g. 2x240 in a NVIDIA GTX 295, but its drawback is the access to memory of such big number of cores. The memory is organized in different types: global, constant, texture and shared, but each type is accessed by using a single port, therefore the serial part and the bottleneck of the algorithms programmed in GPU is the memory access. According the Amdahl's law this is the factor that limits the efficiency of this massive parallel system.

Figure 1. Computational arrangement for fast pattern matching. Data inboxes is the *imm8* value

The main decision in the GPU programming is the design criterion of how memory will be used. We have decided to place the mask data in constant memory and the image and the result data in global memory. The image is constant in the algorithm but it does not fit in the small constant memory of the GPU. The CUDA programming methodology [Nvi09a] allows arranging the processors as a grid of threads. In our problem, the grid of threads is configured by assigning a thread to each result pixel at (x, y) excluding the border band; that is, each thread is involved in the computation of a $R(x, y)$ result. To reduce the negative effects of memory access, the mask data are placed in constant memory because this type of memory is cached and is smaller in size than the mask data. Also, we take advantage of broadcast access to that memory type because for (u, v) values of the mask indexes all the threads access to the same $M(u, v)$, which takes advantage of the broadcast access to constant memory.

In the developed code, the access to global memory is coalescent but unaligned due to the sliding. This would require the access to two consecutive data block in the same half ward, depending on the sliding value of v. To avoid the decreasing in performance of unaligned access, NVIDIA documentation [Nvi09a] suggests the use of texture memory instead of global memory, but we have not experimented any increasing in the performance when allocating the image data in texture memory and the mask in constant memory. At this point, the provided results are the related to the data ***D*** being placed in global memory, with coalescent but unaligned access, and the mask ***M*** being in constant memory with cached and broadcast access. To achieve the computation of the SAD, the unsigned integer version of the CUDA function *sad(a,b,c)* was used, where *a* becomes the value of the sliding image in global memory, *b* the value of the mask in constant memory and *c* the value of the serialized computation of the result.

To increase the efficiency in GPU, streamed calls to memory transfer and kernel launches have been used by splitting the image in several areas, non overlapping in result data and overlapping in image data. The interleaving between data transfer and kernel computation allows hiding the time used in data transfer between device and host. For this streamed asynchronous data transfer, the optional pinned memory allocation was used. The last included improvement is the use of this methodology to feed data and kernel launches in the two GPU contained in the same graphic card.

5. RESULTS

To test the implementations of raw template matching, images of 640×480 pixel have been used. Mask sizes range from 4×4 to 32×32. Both image and mask are single-channel with a pixel data of 8 bit unsigned int. The test computer is a Core2 Quad Q8300 with 4 GB of RAM memory and one NVIDIA GTX 295 graphic card. The Operating System is Windows XP and the code has been written in C++ in Visual Studio. Computational times are obtained by using performance counters of Windows, the reported values are the average on one hundred runs. Table 1 contains the results for four different implementations. The first one is C plain with low performance, but that is used as the baseline to provide the speedup for higher performance

implementations. The second implementation uses SIMD instruction in the 1D matching in which is based in the 2D. Also, for these two implementations the use of multiple cores is included by using OpenMP [Cha08a] parallelism. The 2D matching is implemented by row oriented 1D matching primitive, an *omp parallel for* directive of OpenMP is used for the row loop. The static schedule strategy is used, so the computation of rows is assigned to each core at the thread forking. Four threads are used for this four core system. The speedup (Column Sp) reported is the SIMD implementation with OpenMP in relation to the serial C plain one. Core based parallelism is more effective when 32×32 mask is used because the latencies of the threads forking are less relevant in bigger tasks than in the lowest associated to the small 4×4 mask due to the effectiveness of threads level parallelism is greatly dependent on the computational grain size. A remarkable speedup value of 184 is achieved for 16×16 mask when the four core of the system are used and also their high specialized SIMD vector units.

	1 Core/ 1 Thread		4 Cores / 4 Threads		
Mask	Cp	SIMD	Cp	SIMD	Sp
4x4	11.1	0.7	2.7	0.6	18.5
8x8	33.1	0.8	8.8	0.2	165.5
12x12	66.4	2.4	27.2	0.9	73.8
16x16	110.6	3.4	29.0	0.6	184.3
24x24	231.3	6.5	60.0	1.3	177.9
32x32	388.6	10.5	101.0	2.3	168.9

Table 1. Time in msec. for Template Matching in 640x480 images. C plain (Cp), SIMD and Speedup (Sp) for the 4 Cores case are included.

Table 2 contains the results for the implementations using one GPU. It contains the host to device (HtoD) data transfer, that in this case means, the transfer of mask from host memory to constant memory and the image transfer to global memory. This table also includes the kernel computation and finally the device to host (DtoH) transfer of the result to host memory. The result is coded as unsigned int which is better for the SAD computation but increases the data transfer time. However, this decision is not relevant in bigger mask sizes. Speedup columns are related to the kernel part (Sp1) or the total time (Sp2) over the C plain simple case. Figure 2 shows the graphical representation of these values. In the 4×4 case, data transfer is the critical subtask, while for the 32×32 case the kernel computation is highly more significant than the data transfer.

The final test that has been carried out includes other computational advantages of the GPU. The first one is the use of the second GPU included in the GTX 295 graphic card by means of splitting the whole image in two parts and by loading each part to each GPU. This methodology is extended by splitting the image in many parts and transferring each one to the GPUs in different threads context. This is implemented by thread forking in OpenMP in a number of threads defined by the user and using the *cudaSetDevice* function to select the device. Pinned memory is used to accomplish streamed asynchronous memory transfer and kernel launches. Table 3 contains the results for several mask size, and Figure 3 illustrate the results for 32×32 mask by using synchronous and asynchronous memory transfer. The 2 thread case, which uses a task in each GPU, is the best result followed by the 4 threads, which includes two tasks in each CPU. The asynchronous case that hides part of the memory transfer cost is the best case, but it cannot overtake the SIMD implementation.

Mask	HtoD	Kernel	DtoH	Total	Sp1	Sp2
4x4	0.4	0.3	1.0	1.7	37	6
8x8	0.4	0.7	1.0	2.1	47	16
12x12	0.4	1.5	1.0	2.9	44	23
16x16	0.4	2.7	1.0	4.1	41	27
24x24	0.4	5.8	1.0	7.2	40	32
32x32	0.4	10.1	1.0	11.5	38	34

Table 2. Results in msec. for Template Matching in one GPU. Speedup 1 (Sp1) is Kernel time and Speedup 2 (Sp2) is Total time, both related to 1 Core C plain case.

6. CONCLUSIONS

Modern computer hardware allows the computation of raw template matching in few milliseconds for small mask sizes. The use of ROI can reduce this computational cost to sub milliseconds scale. This fact is an opportunity to revalorize the developing of template matching applications. The use of many cores or many GPUs are different options to consider, but nowadays, the many-core approach, which includes specialized vector SIMD instructions, is more computational efficient because it is high specialized in SAD arithmetic and in-hardware sliding of mask across data array.

Technological improvements are in the line of increasing the number of cores in host. This trend is not well suited for small masks because they do not use effectively the OpenMP parallelism, but in bigger masks they can greatly increase the performance of

template matching procedures. The evolution of GPU architectures, e.g. the new NVIDIA Fermi architecture, will introduce full global memory cached access that can improve the kernel part of the GPU procedure as well as an increase in the number of involves cores. However, this can be improved only in the performance of big mask sizes whereas in smaller ones it will depend on the increase of the bandwidth of memory transfer between the host and the graphic card.

	4x4		16x16		32x32	
Th	S	A	S	A	S	A
1	2.6	2.3	5.1	4.7	12.5	12.1
2	2.4	2.1	3.7	3.3	7.4	7.1
3	5.8	3.6	7.5	5.1	12.6	10.4
4	5.9	3.7	7.3	4.8	11.1	8.7
5	6.2	4.7	9.3	6.0	14.0	10.7
6	6.4	5.0	7.8	6.0	13.8	10.3
7	7.8	5.9	9.5	7.3	14.2	12.0
8	7.9	6.2	9.4	7.3	13.7	11.3

Table 3. Results in msec. for 2 GPUs and OpenMP using different threads number (Th) from 1 to 8. The Sync (S) and Async (A) cases are included for several mask sizes.

7. REFERENCES

[Bra08a] Bradski, G. and Kaehler, A, Learning OpenCV, Computer Vision with the OpenCV Library, O'Reilly, 2008.

[Bru09a] Brunelli, R, Template Matching Techniques in Computer Vision, Theory and Practice, John Wiley and Sons Ltd, 2009.

[Cha08a] Chapman B., Jost, G and van der Pas, R., Using OpenMP Portable Shared Memory Parallel Programming, MIT Press, 2008

[For02a] Forsyth, D.A., and Ponce, J., Computer Vision: A Modern Approach, Prentice Hall, 2002.

[Har01a] Hart, P.E., Duda, R.O., Stork D.G., Pattern Classification, John Wiley and Sons, 2001.

[Int09a] Intel 64 and IA-32 Architectures Software Developer's Manual, Vols 2A-2B, Intel Corporation, 2009

[Kin07a] Kim, H.Y, and Araujo S.A., Grayscale template-matching invariant to rotation, scale, translation, brightness and contrast, IEEE Pacific-Rim Symposium on Image and Video Technology, Lecture Notes in Computer Science, 4872:100-113, 2007.

[Kua07a] Kuah, K, Motion stimation with Intel streaming SIMD extension 4. Technical report, Intel Software Solution Group, 2007

[Luk81a] Lukas B.D. and Kanade, T., An iterative image registration technique with application to stereo vision, Proceeding of the 1981 DARPA Image Understanding Workshop, pp 121-130, 1981.

[Nvi09a] NVIDIA CUDA, Programming Guide, Version 2.3.1, NVIDIA Corporation, 2009.

[Sha01a] Shapiro, L., and Stockman G., Computer Vision, Prentice Hall, 2001.

[Teh88a] The, C.H., and Chin, R.T., On image analysis by the method of moments, IEEE Trans. on Pattern Analysis and Machine Intelligence, 10(4):496-513, 1988.

[Ull04a] Ullah, F., and Kaneko, S., Using orientation codes for rotation-invariant template matching, Pattern Recognition, 37:201-209, 2004.

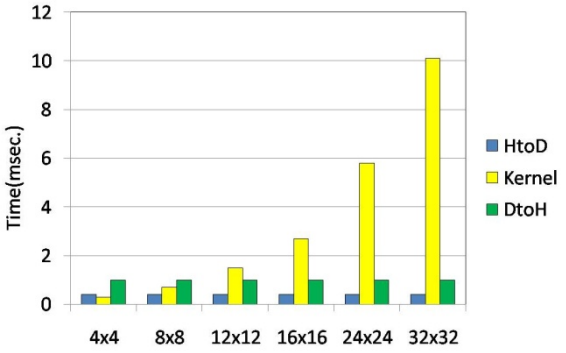


Figure 2. Kernel and Data transfer in GPU, memory copy from host to device (HtoD), device to host (DtoH) and kenel matching

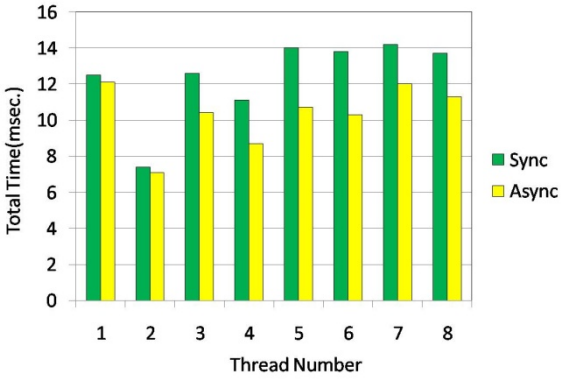


Figure 3. Total time for 2 GPUs and 32x32 mask matching. Asynchronous memory transfer was used with interleave between data transfer and kernel launches.

Approximate reconstruction of meshes after material removing

Félix Paulano	Juan J. Jiménez	Antonio Martínez	Rubén Pulido
Universidad de Jaén Dep. Informática Campus Las Lagunillas 23071, Jaén, Spain	Universidad de Jaén Dep. Informática Campus Las Lagunillas 23071, Jaén, Spain	Universidad de Jaén Dep. Informática Campus Las Lagunillas 23071, Jaén, Spain	Universidad de Jaén Dep. Informática Campus Las Lagunillas 23071, Jaén, Spain
fpaulano@ujaen.es	juanjo@ujaen.es	amalbala@ujaen.es	rpulido@ujaen.es

ABSTRACT

Boolean operations are complex, so it is difficult to perform them in real time. Sometimes, the goal is only to reconstruct the model. In that case, accuracy is not too important and other approaches can be performed. However, the reconstruction of the model must satisfy some requirements like smoothness or velocity. In this paper, a method to reconstruct a model after a cut is presented. This method can be applied to simulate medical procedures, such as the rejection of damaged tissues, or applied to virtual sculpting. A haptic device has been used to test the effectiveness of the method. Tests have shown that the elimination and the reconstruction are performed in real time.

Keywords

Mesh reconstruction, Boolean operation, Material removing, Concavity regeneration, Surgery simulation, Virtual Sculpting, Simulation tools

1. INTRODUCTION

Boolean operations allow performing unions, intersections, differences and other operations between two solid models. This kind of operations is traditionally based on the marching cubes algorithm [NY06]. These operations are very costly because they are extremely complex and its result aims to be exact. For that reason, it is very expensive to use them in a real time application. In simulations, other approaches faster than Boolean operations must be used and accuracy is not usually a key factor.

Our aim is to perform difference operations between two solid models (A-B) in an approximated way. Specifically, the first solid (A) has at least a part slimmer than the second one, and the second solid (B) act as a tool to remove portions of the first solid (figure 1). This is because the solid must be inside the tool so that it can be cut.

This kind of operations is commonly used in virtual simulations, such as virtual surgery or virtual sculpture. In this case, Boolean operations cannot normally be used because they are so costly. New approaches that perform this operation in real time are needed. These new approaches do not have to be exact, but they have to allow real time interaction.

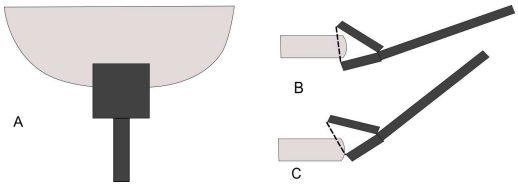


Figure 1. Example of application of the method. A – An aerial view. B, C – Side view of two separate cases.

This paper presents an approach to perform an approximate reconstruction of a mesh. This approach allows realizing a reconstruction that can be applied to real time simulations. The obtained reconstruction is not exact, but in some cases, such as virtual surgery or sculpture, it is an advantage over other approaches. This is because our method obtains a smooth surface after the cut, simulating the real cut of a certain tool with specific types of tissues. In the next section, some works recently published, related to this research area, will be described. Then, steps of the simulation will be enumerated in a general way and

the method will be described in detail. In addition, some special cases will be described. The fourth section will show the simulations that have been performed in order to apply the method. Finally, the simulation results will be presented as well as a brief conclusion.

2. BACKGROUND

In the bibliography, there are some recently works that propose new Boolean operations approaches.

Wang [Wang10] presented a method to perform approximate Boolean operations of two freeform polygonal meshes using Layered Depth Images (LDI). A trimmed adaptive contouring method is developed to reconstruct the mesh surface from the LDI samples near the intersected regions and then suture it to the boundary of the retained surfaces. This method can perform Boolean operations of freeform solids in a few seconds. Jing et al. [JWBC09] proposed an approach to perform Boolean operations on polygonal meshes that can be applied to both closed meshes and open meshes. They use a collision detection algorithm based on OBB trees to speed up the intersection between each two triangles. Then, the intersection region is obtained from the intersected triangles and the intersection segments. Zhou et al. [ZWSW*10] proposed a Boolean operation method based on L-Rep model of 3D entities. The speed of the method is improved by changing the three dimensional spatial analysis into a one-dimensional calculation.

These approaches can perform a Boolean operation in a several tenths of a second. However, it is not enough to apply them in real time simulations. For that reason, other methods that allow performing the approximated operation in real time must be developed.

There are many works that perform a mesh reconstruction using a method based on the Delaunay triangulation [Shewchuk02]. These works perform a homogeneous triangulation. However, our aims are to develop a quick and robust method for simulating a cut so the size and the form of the new generated triangles could be non-homogeneous. Moreover, we know the approximate shape of the resulting mesh before the reconstruction; hence the performance of the method can be improved using that information. Other approaches [Frisken99] propose a linked volume representation that enables physical modeling of object interactions, such us deformations or interactive objects deformations. [NS00] presents an interactive algorithm for an interactive linear FE deformation simulation. Moreover, runtime changes of the mesh can be realized because the process requires no global precomputation.

We propose a method that allows performing an approximate mesh reconstruction after removing triangles. The difference from other approaches is that our method first performs a material removing and then reconstruct the hole of the mesh, obtaining an approximated solution near to the real cut of the tool. The result is a smooth mesh on the border of the operation (on the border of the hole) (figure 2) which is suitable for some type of operations with this type of tools (e.g. for virtual sculpting or surgical simulations). Moreover, enables real time interaction.

3. DESCRIPTION OF THE PROCESS

In order to exemplify, it has been taken into account that the mesh used to remove material is shaped like a cuboid. However, this mesh can be any shape possible, such as a sphere, a cylinder, a star, or otherwise. Hence, the method can be easily applied to other solids. Our approach can be divided in the next steps:

1. Removing triangles. The triangles of the solid A that are inside the cuboid are removed.
2. Transforming the hole. The hole created after removing triangles is transformed in a convex concavity.
3. Projecting the cuboid points. The four cuboid vertices that are nearest to the solid A are projected on it. These four projected points will be used to reconstruct the mesh.
4. Classifying the sutured points. Projected points and sutured points are classified into four quadrants.
5. Generating new triangles. The last step is to reconstruct the mesh using all previous calculations.

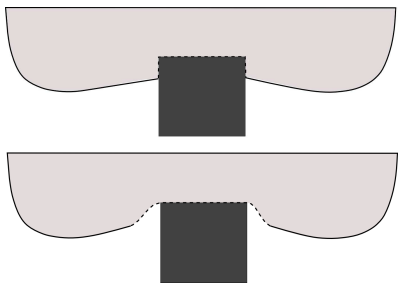


Figure 2. Top, example of a mesh reconstruction using a Boolean operation. Bottom, example of a mesh reconstruction using our approach.

After applying the method, the cut is simulated and the resulting mesh is closed and approximated to the Boolean operation between the trajectory of the tool and the original mesh. In addition, the appearance of the cut is smooth.

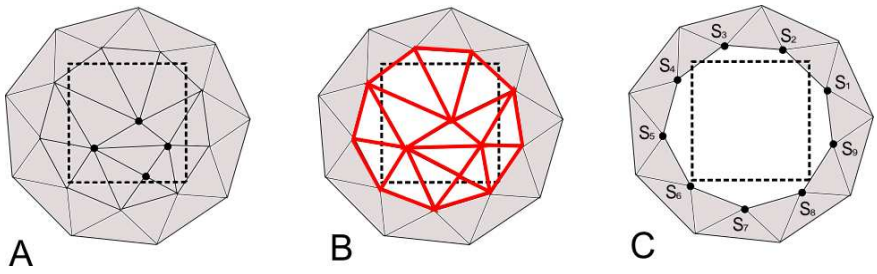


Figure 3. 2D schema of the method used to remove triangles. A – Points to remove. B – Triangles to remove (red). C – Suture Points (s_1, …, s_9)

Removing triangles

Before reconstructing the mesh, triangles of the solid A that are inside the cuboid must be removed. In order to achieve this, triangles that have at least one of its vertices inside the cuboid are deleted.

To simplify this procedure, a spatial decomposition has been performed using a tetra-tree [JFSO06]. This data structure is built in an initial step, so it does not reduce the method efficiency. A tetra-tree is an hierarchical space decomposition defined in the whole space. At its first level, the tetra-tree divides the entire space into eight tetra-cones. In the following levels of the hierarchy, each tetra-cone is homogeneously divided into four new tetra-cones, as shown in figure 4. The tetra-tree is subdivided until reaching one of the following conditions:

- The maximum level of subdivisions is achieved. This level is previously defined.
- The tetra-cone to subdivide has less triangles than a threshold.

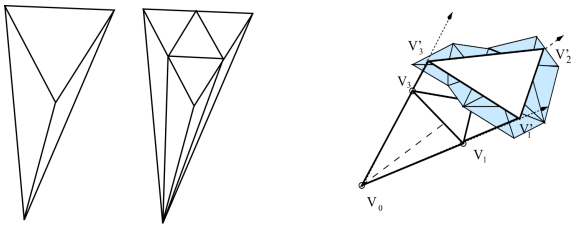


Figure 4. Left, a representation of the division of a tetra-cone. Right, a scheme that represents the bounding tetrahedra associated with a mesh.

Hence, the cudoid only have to interact with the triangles belonging to the tetra-cones where it is situated. The complexity of the calculation associated with the removing is reduced to the tetra-cone space.

This type of spatial decomposition allows us to quickly locate the nearest object part (tetra-cones) where the points are situated. The tetra-tree also allows us to discard triangles for removing in an optimal way, due to the adjustment obtained by bounding tetrahedra [JS08] associated with each tetra-cone. This is represented in figure 4.

The tetra-tree has been chosen because fits the mesh better than other approaches [JS08], such us an octree. In addition, the tetra-tree allows classifying triangles quickly and robustly because is based on barycentric coordinates. While the removed triangles are being determined, the topology of the hole is calculated. In order to achieve this, the triangles that have only a vertex inside the cuboid are used. The two vertices that are outside the cuboid are marked as suture points. In order to simplify the reconstruction, the topology of the suture points is stored, sorting them in opposite counter clock wise. This procedure is shown in figure 3.

Transforming the hole

In this step, the hole created after the elimination of triangles is transformed in a convex concavity using the topological information about its edges. The algorithm can be divided into three parts.

First, the concavities are located (Figure 5). Points that form the hole, which are marked previously as suture points, are studied in groups of three consecutive points. If the sign of the matrix determinant formed by three consecutive points is negative, a concavity is found. During this process, consecutive concavities are grouped into one (e.g. p_1-p_4 in Figure 5).

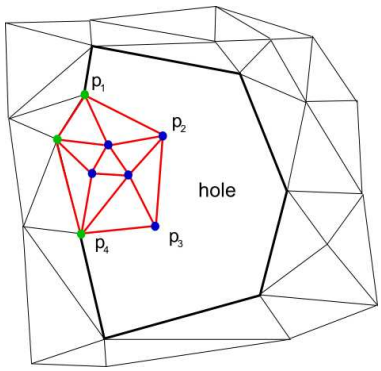


Figure 5. Schema of the method used to eliminate the concavity. Red – Triangles to remove. Blue – Points used as input in the iterations of the algorithm. They are also removed. Green – Points to reconstruct.

Second, once all the concavities are located, they are converted into convexities. First at all, the first and the last point that form the concavity are stored (p_1

and p_4). Then, the remaining points that form the concavity are processed in a loop. This loop is repeated until there are no more input points.

The loop has the following steps:

- Triangles that contain the input points are determined.
- Points belonging to those triangles are studied to check if they are inside or outside the concavity:
    - Points that are inside will be input points in the next iteration of the algorithm.
    - If two points of a triangle are outside the concavity, they will be stored as points to reconstruct and their topology will be saved.
    - Points which have been input points in this iteration will be discarded.

When there are no input points, the loop is finished, obtaining then a set of points to reconstruct. This procedure is shown in figure 5. The first and the last point of the concavity are within the set of points to reconstruct.

Finally, the points to reconstruct are used in groups of two consecutive points to generate new triangles (figure 6). The midpoint of the segment that goes from the first point of the concavity to the last point of the concavity is calculated. This point is used as a third point of each triangle.

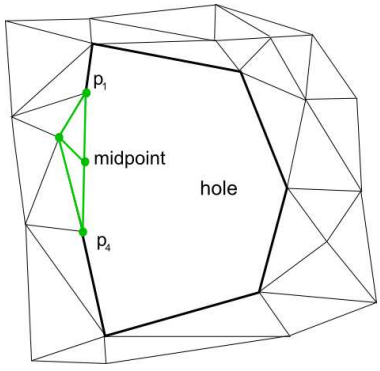


Figure 6. Schema of the hole reconstruction. Green – New triangles added.

The elimination of the concavities in the hole allows simplifying the subsequent generation of new triangles, preventing cross segments. Moreover, the cut obtained is smooth, so some simulations are more realistic than an exact approach.

Projecting the cuboid points

The four points representing the cuboid that are nearest to the solid are used to reconstruct it, so these points are projected on the solid using the algorithm proposed by [JSF10]. This is done to simulate the trajectory of the tool to perform cutting. In order to simplify the procedure, two planes representing the solid are determined. Hence, the points are projected on one of these two planes, instead of the solid.

In the general case (figure 1, B), two planes are calculated. These planes (figure 7) represent the top and the bottom of the solid so they are called upper average plane and lower average plane respectively. A medium plane is used for dividing the triangles in the tetra-cones scope in which the tool is included. Then the triangles in the top and in the bottom are used to obtain two average normals. The upper average plane is defined by the triangles whose normal is similar to the top average normal. On the other hand, the lower average plane is defined by the triangles whose normal is similar to the bottom average normal. In both cases, an error must be considered. Then, the upper points are projected on the upper average plane and the lower points are projected on the lower average plane, as shown in figure 7.

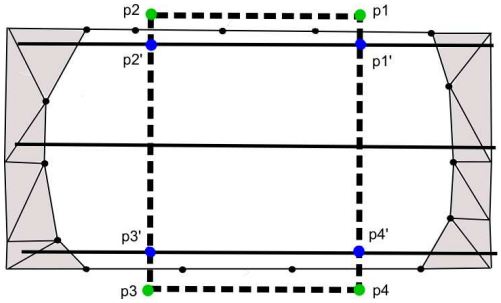


Figure 7. A 2D scheme that shows the projection of the cuboid points (p1, p2, p3, p4) on the upper and lower average planes (p1', p2', p3', p4').

If any of the points cannot be projected on its respective plane, an oblique cut is detected (figure 1, C). In a oblique cut, the part of the object to be cut is not completely within the tool. This is usually because the tool is not aligned with the object. In that case, one auxiliary plane is used to project those points. To calculate the auxiliary plane, some calculations are made in real time. The removed triangles that are situated between the upper average plane and the lower average plane are marked as central triangles. The auxiliary plane is defined by the average point and the average normal of the central triangles.

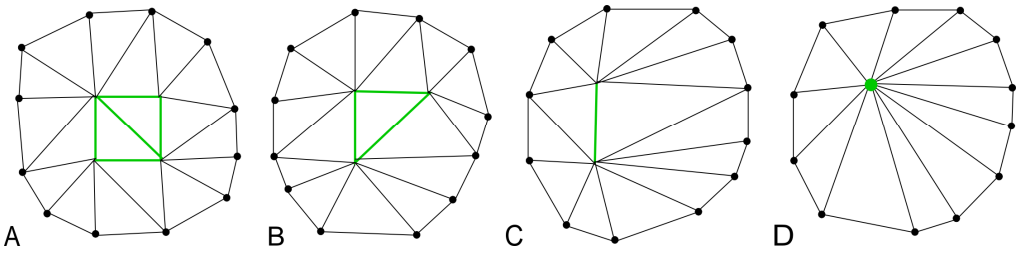


Figure 8. Different cases of generation of new triangles. The procedure depends on the number of projected points that are inside the solid: 4 (A), 3(B), 2 (C), 1(D).

After projecting them, the four projected points are used to reconstruct the solid. However, only the projected points that are inside the solid mesh are used. To check this, the inclusion algorithm by Feito [FT97] is used, because it allows determining if a point is inside a mesh without the need to perform complex calculations, such as solving a system of equations. In addition, to further reduce this problem, the previous spatial decomposition using a tetra-tree [JFSO06] is also utilized.

Classifying the suture points

In order to simplify the reconstruction of the mesh, the suture points and the projected points are classified into four quadrants. The central triangles, which were calculated in the previous step, are also utilized in this procedure. Two perpendicular planes (figure 9) that pass through the average point of the central triangles are calculated. These planes are also perpendicular to the planes forming the cuboid.

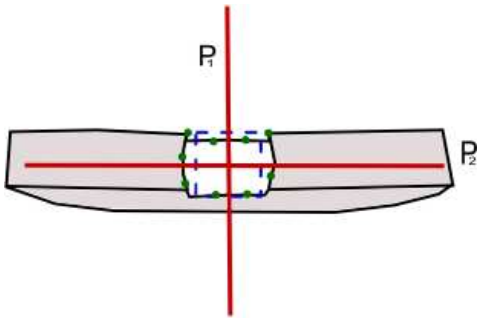


Figure 9. Representation of the two planes (red) forming the quadrants used to divide the suture points (green) and the cuboid vertices (blue).

Both planes divide the previously projected points into four quadrants, assigning each point to a different quadrant. The suture points are also classified using these four quadrants.

Generating new triangles

The last step is to generate new triangles to close the mesh. In order to achieve this, all previous calculations are used.

First, the projected points are used to build a structure. This structure will be used as a patch and it depends on the number of projected points that are inside the solid (figure 8):

- In the case that the four projected points are inside the solid mesh, this structure will be a square.
- The structure will be a triangle if there are three points inside the solid.
- If there are only one or two points inside the solid, it will not build any structure. In this case, the origin of the reconstruction will be the projected point or the segment joining the two projected points respectively.

Second, in each quadrant a sub-mesh is built. In order to achieve this, a triangle is built using each two consecutive points. The projected point assigned to each quadrant is used as the third point in each triangle. If one projected point is outside the mesh, the triangles of its associated quadrant are generated using the projected point of the upper or lower quadrant.

Finally, in order to join the sub-meshes built in each quadrant, new triangles are generated using the projected points and the boundary suture points in each quadrant.

Special cases

In our method, there are some cases that must be treated specially. These cases have been described and can be solved by adapting the general procedure.

In the general case, only a hole is created after removing the triangles. However, several holes may be generated after that (figure 10). In that case, each hole must be treated separately.

This special case is detected in the stage when the hole becomes convex. In this stage, the suture points are read in a loop. The loop finish when all the suture points are read or the first suture point is reached again. If the first suture point is reached again before all suture points are read, the special case is detected. In that case, all unread points are included for processing in a new loop. Each time the first point is founded again, a new hole is detected.

Once the holes are detected, each hole is converted into a convex shape. In order to achieve that, the algorithm explained in the previous section is applied to each hole. Once all the holes become convex, the

four cuboid front points are projected on the planes as in the general procedure (figure 10). Then, all the holes are reconstructed. In order to achieve this, triangles are generated using every two consecutive points belonging to the hole.

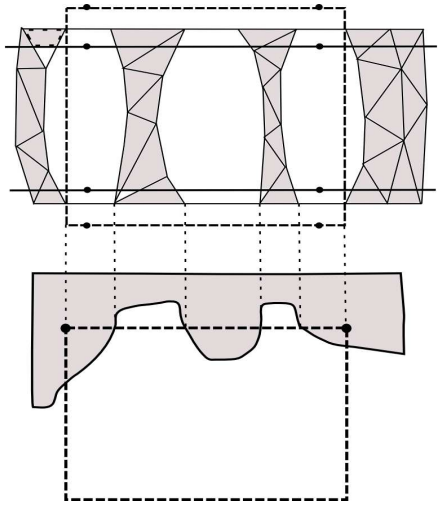


Figure 10. A special case. Three holes have been generated instead of only one. Bottom, an aerial view before cutting.

The average points of the holes are used as the third point of each triangle in the hole reconstruction, as shown in figure 11. Finally, if the two left projected points are inside the solid, the hole nearest to the left is reconstructed using the two left projected points. For that, the approach used in the general case when only two projected points are inside the solid is utilized (figure 11). Similarly, this procedure is repeated with the right projected points and the hole nearest to the right.

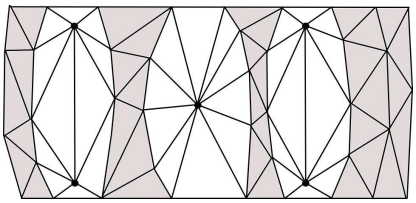


Figure 11. Generation of new triangles in a special case. The central hole is reconstructed using only a point. The others two hole are reconstructed using two projected points.

4. EXPERIMENTAL RESULTS

In this work, a method to realize an approximate reconstruction of a mesh has been implemented. The data structures and algorithms implemented allow simulating the removal and reconstruction of triangles in real time.

In order to prove the method, a virtual meniscus arthroscopy has been simulated. Specifically, a radial injury is treated. This kind of injuries is usually treated removing the damaged area [RBM09] to attempt to keep the stability. Figure 12 shows a representation of the area of the knee treated in the simulation, and a real image of the knee during an arthroscopy. Hence, in the simulation the first solid represents the meniscus and the second solid represents the surgical tool used to remove the damaged tissue.

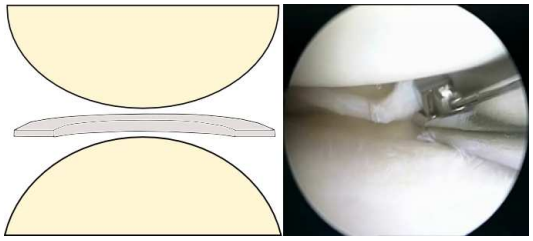


Figure 12. Left, a scheme that represents the area of the knee treated in the simulation. Right, an image of a real meniscus arthroscopy

A haptic device has been used to improve the interaction. Specifically, a Sensable Phantom Omni© and its associated software has been used. The haptic device simulates the surgical tool that is used during the intervention. The surgical tool movement is calculated through the transformation matrix of the haptic device. Moreover, the coordinates of the points that represent the device are calculated to determine the collisions with other elements. The device feedbacks a simple force based on Hooke's law when it detects a collision.

Collision detection

The collision detection [LG98] between the surgical tool and all the triangles that form the meniscus is complex. Moreover, obtaining a real time feedback is complex too, because the refresh rate of the haptic device is 1 kHz [MCL08]. The collision detection involves an intersection algorithm between the surgical tool and the meniscus.

To reduce the problem, the previous spatial decomposition of the meniscus is used, reducing the number of triangles involved in the intersection test. Even reducing the size of the problem through a spatial decomposition, it is still complex to calculate the intersection with the surgical tool. To simplify the collision detection, it only works with a number of representative points of the cuboid and the handle. Then, only these points are classified by the tetra-tree, reducing the collision detection to a point in polyhedra test [JFSO06].

The representative points of the cuboid are their vertices and central points of their faces. To represent the surgical tool handle, taking into account that it is shaped like a small cylinder, a set of points belonging to the cylinder axis are used. These points are shown in figure 13. Once the representative points are calculated, tetra-cones that contain at least one of

them are obtained. These tetra-cones define the space of the meniscus where the intersection will be calculated. To calculate the intersection, the point in polyhedron algorithm by Feito [FT97] has been used.

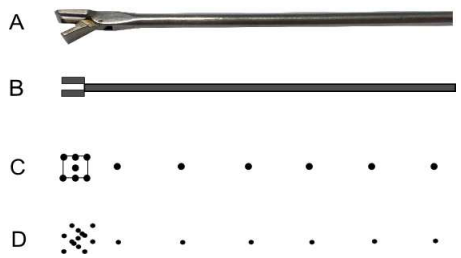


Figure 13. A – The real surgical tool. B – 2D Representation of the surgical tool. C – 2D representation of the tool points used to simplify the collision detection. D – 3D representation of those points.

The use of representative points improves the performance of the simulation, because it allows determining if the surgical tool collides with the elements involved in the simulation in real time, therefore the tool cannot pass through the meniscus.

Mesh reconstruction

One of the device buttons has been used to actuate the tool. If this button is pressed when the surgical tool is close to the meniscus, the tool is actuated, removing a portion of the meniscus. To provide feedback to the user in this action, it has applied a small force. In addition, the tool movement can be restricted as in a real intervention. Therefore, the instrument can be force to only swing through the input. Nowadays, in many cases the repair of radial meniscus injuries is treated by removing the damaged part. Our aims are to apply the proposed method to simulate the rejection of the damaged tissue. Our approach avoids to perform costly Boolean operations between elements [FHK01], giving us a real time interaction. The main advantage of using this approach, instead of Boolean operations, is that our procedure is faster [JWBC09] and response time is a key factor in haptic devices. Boolean operations can obtain more accurately results. However, the accuracy in the meniscus cut is not a critical feature. In figures 14 and 15, results of the simulation are shown. In these images, front and oblique cuts are displayed.

Results

To perform the experiment has been made a subdivision of the mesh that represents the meniscus. This procedure allows us to obtain more complex meshes to measure the performance of the method.

First, it has been measured the time it takes to reconstruct the mesh using different tetra-tree subdivisions. As shown in the table 1, until the fourth subdivision, the more the tetra-tree is subdivided, the better the times obtained are. Moreover, the time it takes to reconstruct without a tetra-tree has also been measured. The results show that the use of our method to remove and reconstruct the mesh, as well as the use of a tetra-tree, enables a real time interaction. Although the refresh rate of the haptic device should be 1 kHz [MCL08], it is impossible to push the button at this frequency, so real time interaction is achieved.

Number of triangles	Time using a level-2 tetra-tree (ms)	Time using a level-3 tetra-tree (ms)	Time using a level-4 tetra-tree (ms)	Time not using a tetra-tree
13618	12,67	8,53	5,87	17,33
25258	29,93	23,33	14,67	73,65
55864	107,07	75,33	42,2	200,67

Table 1. Reconstruction using different tetra-tree subdivisions as well as not using a tetra-tree.

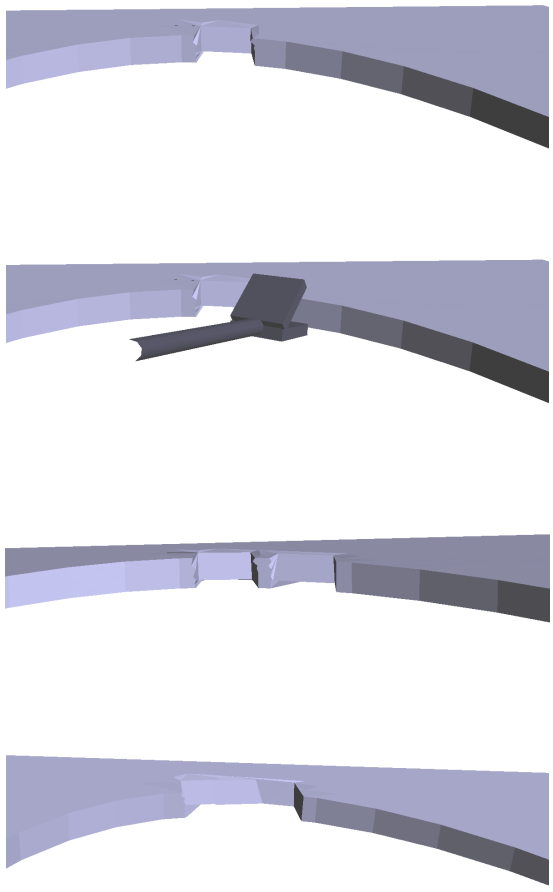


Figure 14. Some images of the simulation. They show front cuts in the meniscus.

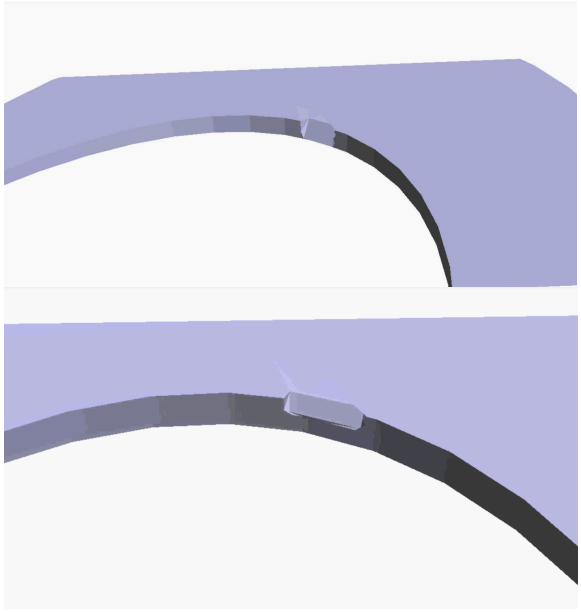


Figure 15. Some images of the simulations. They show oblique cuts in the meniscus.

5. CONCLUSION

In this paper, a method that allows performing an approximate mesh reconstruction after mesh removing has been presented. In contrast to Boolean operations, our method enables real time simulation.

To exemplify the method, it has been considered that the mesh used to remove material is shaped like a cuboid. However, this mesh can easily be any shape. For that, the projected points must be chosen according to the shape of the mesh, instead of the cuboid. The same happens with the points used in the collision detection.

To prove the method, a virtual meniscus arthroscopy has been performed. Specifically, it has been focused on radial injuries. In the future, our method can be applied to perform other simulations, such us other surgery operations or virtual sculpting.

Acknowledgments

This work has been partially supported by the Spanish Ministry of Education and Science and the European Union (via ERDF funds) through the research project TIN2007-67474-C03-03, by the Consejería de Innovación, Ciencia y Empresa of the Junta de Andalucía through the research projects P06-TIC-01403 and P07-TIC-02773, and by the University of Jaén through the research project UJA-08-16-02, sponsored by Caja Rural de Jaén.

6. REFERENCES

[FHK01] Farin, G., Hoschek, J., Kim, M. Solid Modeling. Handbook of Computer Aided Geometric Design. Elsevier Science, 2001.

[Frisken99] Frisken-Gibson, S.F. Using linked volumes to model object collisions, deformation, cutting, carving and joining. Visualization and Computer Graphics, vol. 5, pp. 333-348.

[FT97] Feito, J.R., Torres, J.C. Inclusion test for general polyhedral. Computers & Graphics 21, pp.23-30, 1997.

[JFSO06] Jiménez, J.J., Feito, F.R., Segura, R.J., Ogáyar, C.J. Particle oriented collision detection using simplicial coverings and tetra-trees. Computer Graphics Forum 25, pp.53-68, 2006.

[JS08] Jiménez J.J., Segura, R.J. Collision detection between complex polyhedra. Computers & Graphics 32, vol 4, pp.402-411, 2008.

[JSF10] Jiménez, J.J., Segura, R.J., Feito, F.R. A robust segment/triangle intersection algorithm for interference tests. Efficiency study. Computational Geometry 5, pp.474-492, 2010.

[JWBC09] Jing, Y., Wang, L., Bi, L., Chen, J. Boolean Operations on Polygonal Meshes Using OBB Trees. Environmental Science and Information Application Technology, vol.1, pp.619-622, 2009.

[LG98] Lin, M.C., Gottschalk, S. Collision detection between geometric models: A survey. In Proc. Of IMA Conference on Mathematics of Surfaces, pp37-56, 1998.

[MCL08] Ming, C., Lin, M.O. Haptic Rendering: foundations, algorithms, and applications. A K Peters, Ltd, 2008.

[NS00] Nienhuys, H., Stappen, F. Combine finite element deformation with cutting for surgery simulations. In EUROGRAPHICS 2000.

[NY06] Newman, T.S., Yi, H. A survey of the marching cubes algorithm. Computers & Graphics 5, vol.32, pp.854-879, 2006.

[RBM09] Richmond, J.C., Bono, J.V., McKeon, B.P. Knee Arthroscopy, 2009.

[Shewchuk02] Shewchuk, J. R. Delaunay refinement algorithms for triangular mesh generation. Computational Geometry 1-3, vol.22, pp.21-74, 2002.

[Wang10] Wang, C. Approximate Boolean Operations on Large Polyhedral Solids with Partial Mesh Reconstruction. Visualization and Computer Graphics, no.99, pp.1-10, 2010.

[ZWSW*10] Zhou, L., Wang, D., Sheng, Y., Wang, Y., Li, P. Three dimensional Boolean operation based on L-Rep model. Geoinformatics, 2010 18th International Conference on, pp.1-6, 2010.

Novel Trilateral Approach for Depth Map Spatial Filtering

Alexander Voronov
Moscow State University
Graphics & Media Lab
avoronov@graphics.cs.msu.ru

Dmitriy Vatolin
Moscow State University
Graphics & Media Lab
dmitriy@graphics.cs.msu.ru

Maxim Smirnov
YUVsoft Corp.
ms@yuvsoft.com

ABSTRACT

In this paper, we present our approach for spatial filtering of depth map extracted from camera motion. An original depth map may have some artifacts owing to imperfect motion estimation. Our goal was to make the depth map uniform in smooth areas and to refine object boundaries without blurring edges. To solve this problem we propose the trilateral filter, whose convolution kernel is composed of a distance kernel, a color-based kernel and a depth-based kernel. Experiments demonstrate that this approach yields rather good results. Also, we compare our results with those of a typical bilateral filter.

Keywords

depth map, disparity map, trilateral filtering, spatial filtering, post-processing, 3D video

1 INTRODUCTION

One of the widely used methods of creating 3D video involves changing the image parallax using a depth map. This process requires information regarding the distance between the camera and the objects in the scene. A depth map is a visualization of that distance for every pixel in the image: more-distant spots are represented using a darker color. Generally, the problem of creating of depth map from a single image is insoluble. So, until recently, depth maps were painted manually by stereo artists and composers in most cases – a task that required much time and money. But in some cases, depth maps can be created on the basis of information from a scene. Some such approaches apply machine learning algorithms to extract the information from a rather large set of images in different scales [Sax06]. Also, [Zhu09] proposes an approach that uses the fact that the camera is typically focused on foreground objects, so that objects have sharp edges: with increasing distance from the camera, object boundaries become blurrier. Another approach is to restore depth using the geometric properties of a scene: for example, by taking into account the vanishing point, horizon line, vertical lines and so on.

This technique is presented in [Bat04] and [Jun10]. For scenes with camera motion we can create a depth map by applying an optical flow algorithm and analyzing how objects are moving in a scene. For example, if the camera is panning, an object's displacement in a given frame relative to the previous frame depends on that object's distance from the camera. This approach is described in [Pou10] and [Kim07].

Application of an optical flow algorithm supposedly yields the highest quality depth map estimation using camera motion. But the results of the algorithm at that stage may be not good enough for several reasons. First, it is impossible to accurately determine optical flow for two frames in regions of opening and occlusion that appears when objects are in motion. In such regions depth can not be estimated correctly. Second, it is impossible to detect true motion in smooth areas, particularly in case of noisy video. Third, considering the high computational complexity of this stage, we must often sacrifice optical flow quality to increase processing speed; this affects final results. So, some postfiltering is required to reduce errors in a depth map or to make them less visible in the final result.

Such postfiltering can be performed using some variations of simple Gaussian smoothing [Zha05] or using more-complex filtering: for example, bilateral filtering [Cha09]. To address the problem of stereo correspondence this approach can be extended to use multilateral filtering, particularly with a left-right consistency metric, which makes it more robust. Details of this approach are presented in [Mue10] and [Jac10]. Other approaches under active development

reduce the problem of depth estimation to a matter of energy optimization for the whole frame, a process that requires extensive processing time but produces better quality for the final results [Zha08].

2 PROPOSED METHOD

Our work involves spatial filtering of a depth map that was estimated using camera motion for single-view video. Our approch can be applied to depth maps generated by any other method, however, because the filtering does not use any additional information from the scene.

To suppress artifacts, we propose trilateral filtering. The convolution kernel is built for every pixel and is composed of the following components: Gaussian kernel G with a specified radius, matrix $I(x,y)$ based on the photometric difference between the current pixel with coordinates (x,y) and neighboring pixels in the source image, and analogous matrix $D(x,y)$ calculated for this pixel using a depth map.

G responds to the distance from the current pixel being processed: the farther the pixel is from the center the lesser influence it has on the result. Weights $i(x,y)$ in image-based component are linearly dependent on the difference between the central pixel and other pixels:

$$i(m,n) = \\ = \begin{cases} \frac{th_{color} - IDiff_{xy}(m,n)}{th_{color}}, & IDiff_{xy}(m,n) \le th_{color} \\ 0, & IDiff_{xy}(m,n) > th_{color} \end{cases} \quad (1)$$

where $IDiff_{xy}(m,n) = (|red(m,n) - red(x,y)| + |green(m,n) - green(x,y)| + |blue(m,n) - blue(x,y)|)/3$, $m \in [x-r, x+r], n \in [y-r, y+r]$.

Depending on the input data, the color difference may be calculated in another way: for example, as the maximum absolute difference for the color components or as the absolute difference between the average values of the color components. But the results and processing speed vary just slightly so we selected the mean absolute difference as the more general approach. The parameter th_{color} is set accordingly to the source image's noise level and contrast range.

For the depth-based component $D(x,y)$, linear dependence is not applicable. In some models depth has only a few grades, so an error in one depth grade may yield too large a color range. Also, we chose to penalize large differences in depth, so we used a logistic function, and we calculated weights for depth-based component in the following way:

$$d(m,n) = 1 - \frac{1}{1 + e^{-t \cdot DDiff_{xy}(m,n) + 6}} \quad (2)$$

where $DDiff_{xy}(m,n) = |D(m,n) - D(x,y)|$, $m \in [x-r, x+r], n \in [y-r, y+r]$ and t is a parameter that influences on the acceptable deviation of the depth value from value in the central pixel. The constant 6 is based on the properties of the logistic function: the value of the function for arguments greater than 6 is very close to zero.

When we take into account information from a rough depth map, a problem may crop up. All the artifacts in the depth map will influence the depth component in the convolution kernel, and consequently, they will influence the final result. So, to calculate weights in the depth component, we need a depth map that is largely free of artifacts but that is not blurred, having strong edges. To solve this problem we used bilateral filtering with an adaptive threshold. If enough pixels of the same color are near the current pixel, we set the threshold low to preclude using pixel from another depth level and blurring of an edge. But when there are few similar pixels, we set the filtering strength high enough to suppress the artifacts. We choose the filtering radius according to the size of the artifacts that we want to suppress.

Then final convolution kernel $K(x,y)$ is calculated as the element-wise product of matrices G, $I(x,y)$ and $D(x,y)$.

$$k(m,n) = g(m,n) \cdot i(m,n) \cdot d(m,n) \quad (3)$$

The resulting pixel value in the filtered depth map is:

$$r(x,y) = \frac{\sum_{m=x-r}^{x+r} \sum_{n=y-r}^{y+r} k(m,n) \cdot z(m,n)}{\sum_{m=x-r}^{x+r} \sum_{n=y-r}^{y+r} k(m,n)}, \quad (4)$$

where $z(m,n)$ is the pixel in the estimated depth map.

Compared with bilateral filtering, the trilateral approach has some advantages. If we ignore information from the source image, we are only blurring a depth map and are not really enhancing it. But if we ignore depth when building a convolution kernel, we may blur a boundary between two objects of the same color. Also, when we use only color component we may obtain the wrong thin depth flows on boundaries, since boundary colors are usually the average of the objects they divide. An example of flows artifact is presented in Figure 1.

3 RESULTS

Figures 2, 3 and 4 show the results for the proposed method as well as a comparison with the bilateral approach. This method outperforms image-based bilateral algorithm in preserving the boundaries of objects detected by optical flow. Also, it produces smoother depth in uniform areas compared with the depth-based bilateral approach.

a) Source image b) Estimated depth map

c) Result of bilateral filtering d) Result of trilateral filtering

Figure 1: Example of flow artifact for the bilateral filter, sequence "Road", frame 29

4 FUTURE WORK

In the short term, the authors plan to better integrate data from optical flow algorithm into postfiltering algorithm; this integration which will improve the restoration quality for small details and will also allow as to obtain a confidence measure for each pixel. Using this measure, we will be able to estimate the probability of that artifacts will appear in certain regions and allowing us to achieve better results.

Another direction in the algorithm's development is use of temporal data from previous and subsequent frames compensated according to optical flow. This approach will significantly increase computational complexity but should improve the depth map's temporal stability and improve details in a frame.

Also we plan to use the source image and optical flow data to extract separate objects as structural units for more precise processing of object boundaries.

5 CONCLUSIONS

In this paper, we proposed an algorithm of trilateral postfiltering for depth maps created from camera motion. We compared this algorithm with other approaches, and we described and demonstrated the relative advantages of our approach. After reviewing potential problems that can appear when using an inaccurate depth map for calculating the convolution kernel, we described our method of solving these problems. Lastly, we described our intended directions of future work.

ACKNOWLEDGMENTS

This research was partially supported by grant 10-01-00697-a from the Russian Foundation for Basic Research.

REFERENCES

[Sax06] Ashutosh Saxena, Sung H. Chung, *Learning Depth from Single Monocular Images*. Advances in Neural Information Processing Systems, 2006.

[Zhu09] Shaojie Zhuo, Terence Sim, *On the Recovery of Depth from a Single Defocused Image*. Proceedings of International Conference on Computer Analysis of Images and Patterns (CAIP), vol. 5702/2009, p. 889-897, 2009.

[Bat04] S. Battiatoa, S. Curtib, M. La Casciac, M. Tortorac, *Depth-Map Generation by Image Classification*. Proceedings of SPIE, vol. 5302, 95, 2004.

[Pou10] Mahsa T. Pourazad, Panos Nasiopoulos, Rabab K.Ward, *Generating the DepthMap from the Motion Information of H.264-Encoded 2D Video Sequence*. EURASIP Journal on Image and Video Processing, Volume 2010.

[Kim07] Donghyun Kim, Dongbo Min, Kwanghoon Sohn, *Stereoscopic Video Generation Method Using Motion Analysis*. Proc. of the 3DTV Conference, p. 1-4, 2007.

[Zha05] Zhang, L., Tam, W. J., *Stereoscopic Image Generation Based on Depth Images for 3D TV*. IEEE Trans. on Broadcasting, vol. 51, pp. 191-199, Jun. 2005.

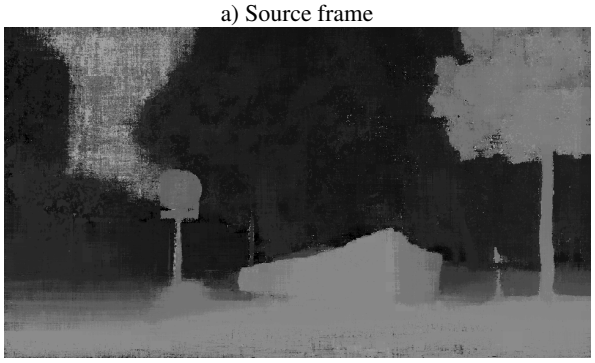
[Cha09] Chao-Chung Cheng, Chung-Te Li, Po-Sen Huang, Tsung-Kai Lin, Yi-Min Tsai, and Liang-Gee Chen, *A Block-based 2D-to-3D Conversion System with Bilateral Filter*. International Conference on Consumer Electronics, p. 1-2, 2009.

[Zha08] Zhang, G., Jia, J., Wong, T., Bao, H., *Recovering Consistent Video Depth Maps via Bundle Optimization*. Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, p. 1-8, 2008.

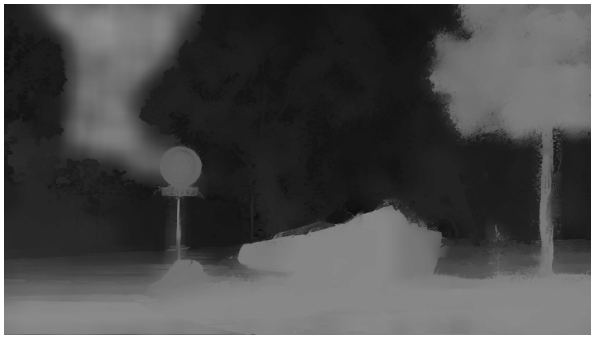
[Mue10] Mueller, M., Zilly, F., Kauff, P., *Adaptive cross-trilateral depth map filtering*. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), p. 1-4, 2010.

[Jac10] Jachalsky, J. Schlosser, M. Gandolph, D., *Reliability-aware cross multilateral filtering for robust disparity map refinement*. 3DTV-Conference: The True Vision - Capture, Trans-

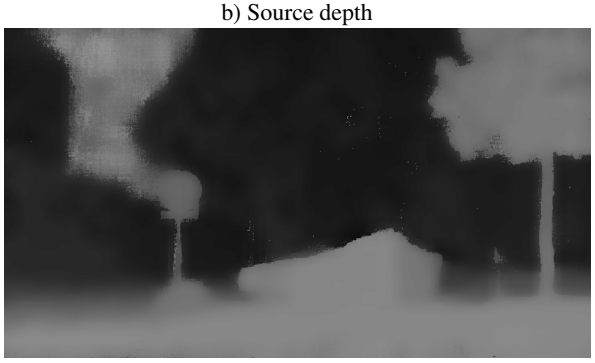
a) Source frame



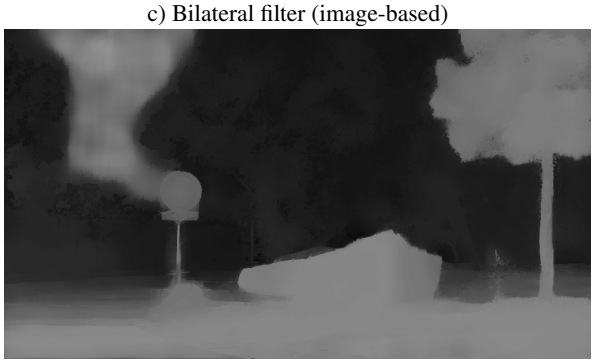
b) Source depth



c) Bilateral filter (image-based)



d) Bilateral filter (depth-based)



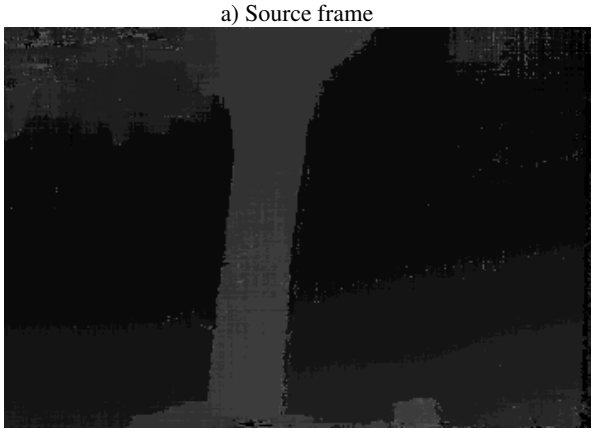
e) Trilateral filter

Figure 2: Comparison of different filtering methods for “Road” sequence, frame 29.

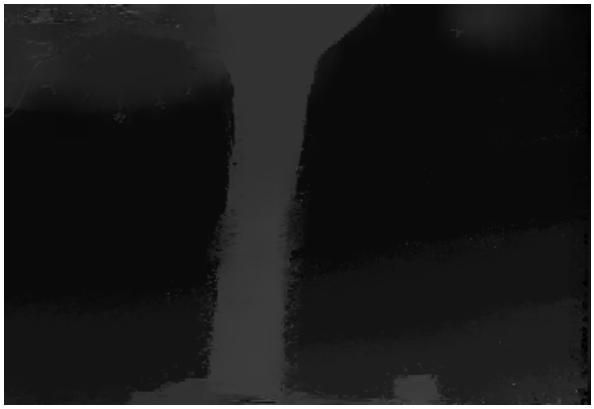
mission and Display of 3D Video (3DTV-CON), p. 1-4, 2010.

[Jun10] Jae-Il Jung, Yo-Sung Ho, *Depth map estimation from single-view image using object classification based on Bayesian learning*. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), p. 1-4, 2010.

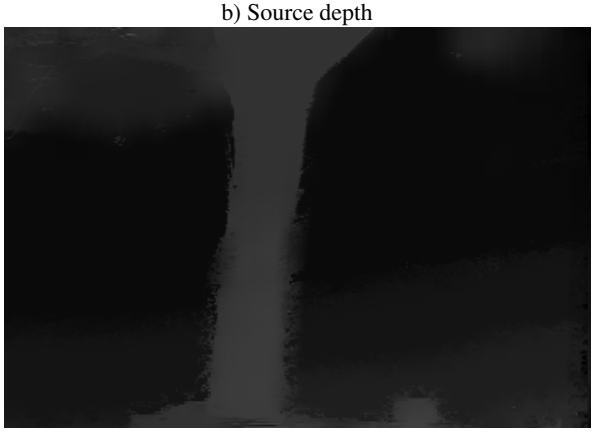
a) Source frame



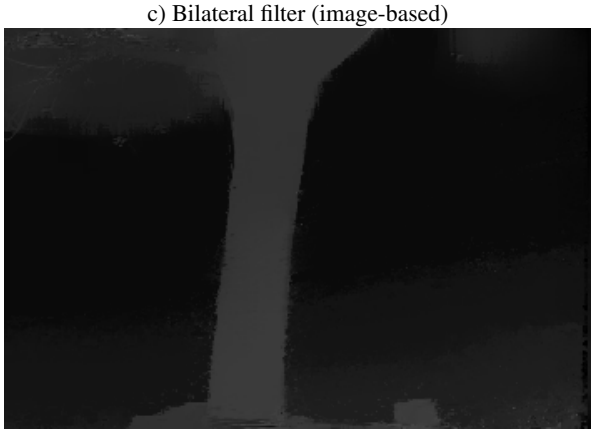
b) Source depth



c) Bilateral filter (image-based)



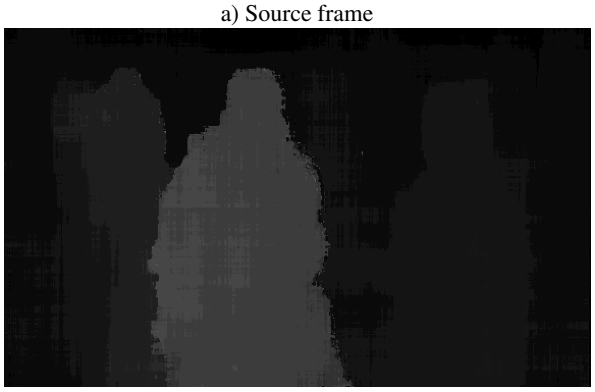
d) Bilateral filter (depth-based)



e) Trilateral filter

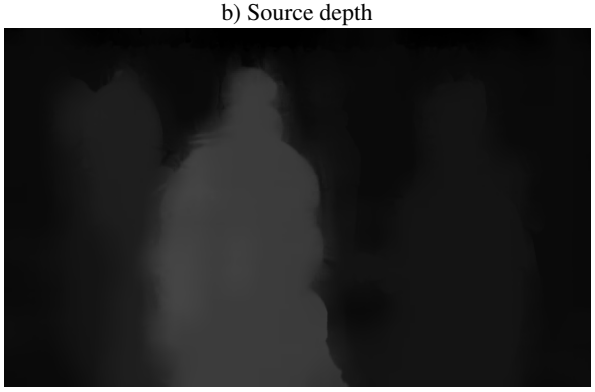
Figure 3: Comparison of different filtering methods for “Garden” sequence, frame 18.

a) Source frame

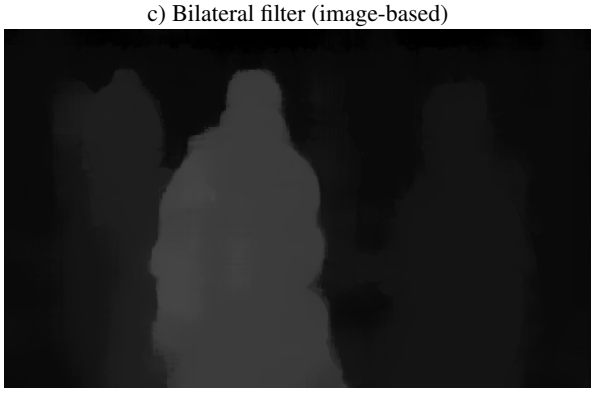


b) Source depth

c) Bilateral filter (image-based)



d) Bilateral filter (depth-based)



e) Trilateral filter

Figure 4: Comparison of different filtering methods for “Warrior” sequence, frame 17.

Network Protocols for Applications of Shared Virtual Reality

Jiri Hnidek

Technical University of Liberec
Studentska 2
Czech Republic 461 17 , Liberec

jiri.hnidek@tul.cz

ABSTRACT

Files are usually used for exchange of 3D data between graphical applications, but this approach is not feasible for applications of shared virtual reality. Thus a network protocol is used for this purpose. Two antithetical requirements are claimed for such protocol. Protocol has to be partially or completely reliable. Neither delay jitter nor too high delay are acceptable. This paper explains and analyzes new version of protocol called Verse. This improves shortcomings found in UDP, TCP, SCTP and DCCP transport protocols. The Verse protocol was designed for sharing 3D data between applications of shared virtual reality. This paper also contains results of experiments comparing suitability of network protocols for application of shared virtual reality.

Keywords

Shared Virtual Reality, Network Protocol, Transport Protocol, Delay, Delay Jitter, Verse

1. INTRODUCTION

Applications of shared virtual reality (ASVR) require transferring of information (e.g. position of avatar [KCJ0]) with small delay and delay jitter, because flickering of object motion is disruptive for an observer of virtual reality. Let consider situation, when users of ASVR cooperate in this environment and try to create large scene (e.g. city with buildings). When users create these objects, then movements of all entities (objects, vertexes) is unpredictable. Each user should see what other users are doing in real-time and movement of shared entities should be smooth as much as possible. Many users of ASVR can create large traffic. Moreover some activities of users can cause burst traffic (uploading of existing object, sculpt painting, etc.).

UDP protocol is usually used for sending real-time data. On the contrary TCP is usually used for transferring static 3D data, because it is reliable stream protocol. When users of ASVR want to edit shared geometry and topology of 3D objects, then partial reliability as well as low latency is required.

It will be proved that any transport protocols as is can not meet those needs. It will be shown that new Verse protocol can effectively meet both needs.

2. CONDITIONS OF EXPERIMENTS

A special client-server application was developed for testing all above network protocols (Fig. 9). Network protocols were tested in real network environments, but comparison of protocols required different approach.

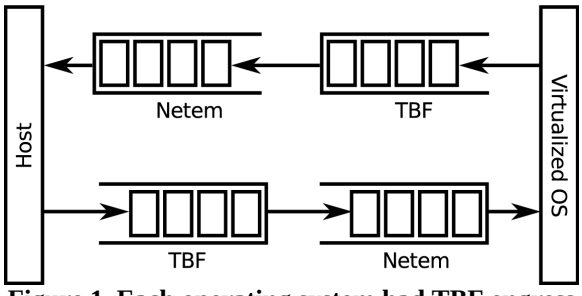


Figure 1. Each operating system had TBF engress qdisc and Netem ingress qdisc. Connection of Netem and TBF qdisc allowed to simulate real network conditions.

It was necessary to set exact parameters of link between client and server applications for tested protocols. For this reason the server application run on virtualized Linux operating system and the client application run on host. Virtual link between host and virtualized OS was modified with Linux traffic

control to simulate real network conditions . The Netem [Hem05] queueing disciple (qdisc) was used for setting delay and delay jitter. The TBF qdisc was used for setting limited bandwidth. It is important to note, that unmodified link between host and virtualized OS had delay 0.5 ms and average delay jitter was 0.03 ms. Configured values of delay and delay jitter were 10 times higher than values on unmodified link. This delay simulated local are network. The bandwidth of the link was limited using TBF qdisc to 256 kb/s. The MTU of link was 1500 B.

3. METHODS OF EXPERIMENTS

Particle system was used to simulate wide range of users working with 3D data. Two particle systems were pre-generated. The particle system containing 100 particles was used for testing network protocols at modified virtual link, because it generated decent traffic and visualization. The particles system with 1000 particles was tested in real network environment. The particle system is simple simulation of bouncing balls, but it is assumed that movement of particles simulates some type of unpredictable movement and thus only position of particles was sent through network. Graph at Fig. 2 shows time slope of particles in scene.

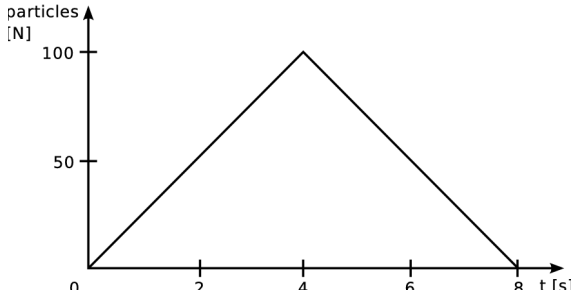


Figure 2. Time slope of particles in scene. Duration of pre-generated particle system was 8 seconds using 25 frames per seconds (FPS).

The particle systems were generated with 25 frames per second (FPS). The illusion of slow motion depends on many factors (resolution, distance of user from the screen, etc.). In the worst case this illusion is broken, when delay of received particle is longer then 40 ms. Such particle is visualized as problematic delay.

The main communication between client and server is started by client application by a request to send particles. When server receives client request, then it starts to send positions of all moving particles every 40 milliseconds back to the client. The PMTU is used during this simple handshake to discover maximal size of packet. The position of each moving particle is added to the packet in a simple message containing particle id, frame number and vector of position. When there are no other particles that could be added to the packet or packet is full, then packet is send to the client. The client application tries to receive packets with positions of particles and it visualizes differences between received and pre-generated particle system. First 10 received blind packets are used to compute average delay between client and server before real data transfer.

4. TESTING OF TRANSPORT NETWORK PROTOCOLS

UDP and TCP

UDP [Pos80] was the first tested protocol. UDP is unreliable datagram protocol widely used in gaming applications for its low latency. UDP is not congestion aware and generated traffic can cause congestion collapse. We can see at Fig 4 that particles transported with UDP had low delays except the period, when congestion occurred. Average delay was bigger than 40 milliseconds and packet loss was visually noticeable during connection. On the other hand tests proved that using pure UDP in ASVR is not feasible, because lost data are not resent and it leads to inconsistency of shared data in ASVR. UDP protocol could be used for ASVR, but re-sending of lost packets has to be solved at the application layer.

Contrary TCP protocol [Pos81] is not widely used in gaming applications, because of the consequences of its reliability mechanism. When one single packet is lost, then proceeding of all following packets is blocked until the lost packet is resent as we can see at Fig 5. Such behavior lead to a sudden stop of motion and high delay up to 1 second. Tests of TCP protocol proved that using TCP in ASVR is possible only in situations where bandwidth is bigger than the highest generated bitrate, there are no other concurrent transmission and RTT is much smaller than 1/FPS. It is usually not possible to guarantee such conditions in real networks and thus re-transmission of lost data leads to very big delays.

Al-Regib and Altunbasak proved in [ARA04], that combination of UDP and TCP connection can be effectively used for streaming of large 3D data sets. This approach can be also used in ASVR for loading large data sets, but it does not solve all specific problems of ASVR, where many users share the same 3D data set.

We can see at Fig. 4 and Fig. 5 that tests of UDP and TCP protocols on real network produced similar results as tests at modified virtual link.

SCTP

SCTP [Ste07] is a modern message based transport protocol. It can act as reliable or partially reliable protocol. SCTP does not provide reliable order delivery, because it is based on message. When reliable variant of SCTP is used, then there is no need to wait for re-transmission of previous lost packets. This feature could be considered as great

benefit for ASVR, because there is smaller average delay of received particles. On the other hand this caused flickering of received particles. Someone can argue that this is visually more confusing than big delays of TCP. Flickering could be theoretically removed by adding a time stamp to each message, but re-transmission of obsolete particle position is not effective approach. Every re-transmission of obsolete particle position makes congestion worse. We can see at Fig. 6 that average delay of reliable variant of SCTP is bigger than average delay of TCP.

Partial reliable variant of SCTP can specify time to live (TTL) of each message. It means that sender tries to re-transmit lost message only for specified time. When the TTL of message is reached, then unsent message is dropped. Partially reliable variant of SCTP removed flickering of received particles, when TTL was smaller then 0.5/FPS, but important feature of reliability was lost. Using partially reliable variant of SCTP in ASVR is not acceptable for the same reason as pure UDP protocol. The results of tests (Fig. 7) gave similar result as test of UDP protocol.

DCCP

DCCP [KHF06] is a congestion friendly unreliable datagram protocol. Congestion control of DCCP protocol should be better than congestion control implemented at application layer on top of UDP, because DCCP can send ECN capable packet that helps to detect congestion without dropping packets. Tests of DCCP protocol proved that implementation of this protocol in Linux is not ready yet for practical deployment. When some error occurs (e.g. many packets loss), then memory of machine can go out of the limit [LF09]. Thus results of DCCP test were not comparable with tests of other transport protocols.

5. VERSE PROTOCOL

All tested transport protocol failed in some way. Unreliable or partially reliable protocols do not resend lost packets. Reliable protocols try to resend all lost data and it causes high delays. Verse protocol uses different approach, because it tries to resend only actual data and obsolete data are dropped.

Verse protocol [BSS06] [SB07] is an application protocol designed for sharing 3D data in ASVR. It uses UDP protocol as a transport layer. UDP it is widely used datagram protocol and it allows implementation of own effective resend mechanism at application layer.

Principles of Verse Protocol

The Verse protocol uses client-server architecture. It means the Verse server holds data and distributes changes of shared data between connected clients. For example, when a client sends a message containing new position of the object to the server, then server changes local position of the object and re-transmits this change to all clients interested in this object. From this point of view the Verse protocol behaves like a network protocol used in gaming applications [WCC+09]. The Verse protocol allows much more. Applications can share not only the object transformations but also geometry and topology of objects, materials, textures, UV coordinates etc. On the other side, the Verse protocol does not allow to use multicast connections because each client is interested in different set of objects.

Resend Mechanism of Verse Protocol

Basic principles of resend mechanism will be described on the example (Figure. 3). It is assumed, that the Verse packet with ID = 31 contained information about object position. This packet was sent from the sender to the receiver. The receiver received this packet and sent the acknowledgment packet back to the sender. The sender could be client and receiver could be server and vice versa.

After a while the sender sent new packet. This packet had ID = 32 and contained new position of the object. This packet was lost. The receiver detected this loss, when the packet with ID = 33 was received, because the receiver expected packet with ID = 32. After this, the receiver sent acknowledgment containing information about reception of packet 33 and loss of the packet 32. The next method of detection of packet loss is detection by sender using timeouts.

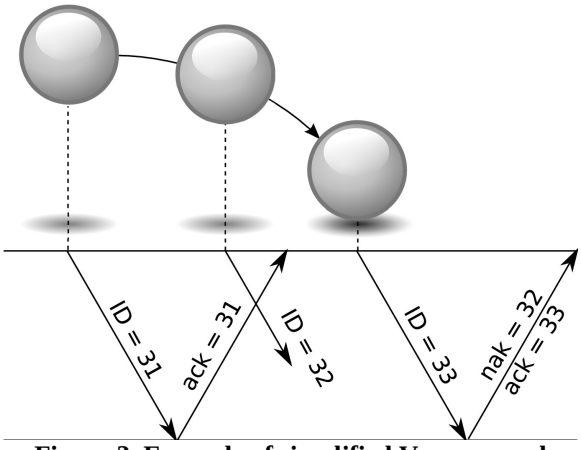


Figure 3. Example of simplified Verse resend mechanism. Each packet sent from the sender has unique ID and it contains position of sphere object. Acknowledgment packet sent from the receiver to the sender contains positive or negative acknowledgment of received packet.

Let's assume, that this acknowledgment was received by the sender. How the sender processed this acknowledgment? If the sender resent content of the lost packet 32, then the receiver would receive obsolete position of the object and it would lead to inconsistency of shared data. On the other side the lost packet could contain some useful and still valid information (for example, information about position

of some other object, command to delete an object etc.). Therefor sender must pick non-obsolete data from lost packet and pack them to a new packet.

It is important to note, that the most of packet loss is caused by congestion in the network. Because network equipments try to use fair scheduling for data flows, then most of congestion is caused by traffic from the sender. Thus most of the packet loss could be effectively detected by methods described above.

If packet loss was detected only by sender using timeout, then this behavior would lead to high delays. Let's consider that retransmission timeout interval (RTO) is computed using the following formulas. Smoothed RTT (SRTT) is computed with:

$$\alpha \cdot RTT + (1-\alpha) \cdot SRTT \rightarrow SRTT \quad (1)$$

where RTT is round trip time measurement from the most recently acknowledged payload packet. The RTO is then:

$$RTO = \beta \cdot SRTT \quad (2)$$

Suggested value of constant α is 0.9 and suggested value of constant β is 2. RTT of packet could be in range of 1-100 ms in real network environment. If SRTT is 10 ms, then not-lost packets are delivered with average delay 5 ms and lost packet would be delivered with average delay 25 ms. Proposed approach used in Verse protocol allows to deliver lost packet with average delay 15 ms. If 3D scene is visualized with 60 FPS, then delay between two frames is 16.7 ms. It is obvious, that Verse protocol has high chance to resend lost packet just in time.

If TCP was used on transport layer, then the packet loss would be solved very ineffectively from our point of view. If the packet 34 was lost, then processing of all following packets would be suspended until content of the packet 34 would be delivered. Such behavior had negative effect on fluency of particle movements in tests of TCP protocol.

The real Verse protocol is more complicated, then example described above. Verse uses two types of packets. Payload packets contain payload data. Acknowledgment packets contain acknowledgments of payloads packets.

Each payload packet has unique ID (Payload ID). A sender increments the counter of sent packets every time it sends a payload packet. When the counter reaches value 2^{32} , then the counter is reset to zero value.

When payload packet is received, then receiver sends an acknowledgment packet to the sender. This acknowledgment packet has unique ID (AckNak ID) and it contains at least one message with the acknowledgment of the received payload packet. This packet could contain more acknowledgment messages (will be described later). The uniqueness of AckNak ID is guaranteed by the same mechanism as in the case of Payload ID. The receiver should send at least one acknowledgment packet for two received payload packets. The receiver should decrease or increase the ratio of acknowledgment packets, when sender detects acknowledgment packet loss. Thus the sender negotiate the ratio of acknowledgment packets.

Negative acknowledgment informs the sender, that one ore more packets were lost. The receiver detects packet loss, when expects receiving of payload packet with ID = N , but payload packet with ID > N is received. The host sends an acknowledgment packet containing all the ACK and NAK messages from the previous acknowledgment packets and following sequence:

$$nak(N), \ldots, nak(ID-1), ack(ID) \quad (3)$$

When delayed packets (considered as lost) are received, then it is possible to process non-obsolete data from these packets, but it is easier to drop them.

Delivery of acknowledgment packet is uncertain, because an unreliable datagram protocol on the transport layer is used. Therefore, probability of delivery of an acknowledge packet to other side must be maximized. All the ACK and NAK messages from previous acknowledgment packets are added to further packets, including payload packets. It is clear, that adding the ACK and NAK messages to packets should be limited somehow. The ACK and NAK messages could not be added to the packet infinitely, because traffic with low packet loss and high delay could produce long sequence of ACK and NAK messages. In this manner ACK and NAK messages would fill the whole packet in a short time.

To avoid infinite increase of the ACK and NAK messages, acknowledgment of acknowledgment has to be added to the Verse resend mechanism. The ID of the last acknowledged payload packet is added to the packet sent to the peer. This ID is called Ank ID. When the receiver receives such packet, then it is necessary to send only ACK and NAK messages for payload packets greater than Ank ID. The sender sends packets with the Ank ID until a newer acknowledgment packet is received.

The next mechanism of limiting sequence of ACK and NAK messages is compression of this sequence. Let's consider the following sequence of ACK and NAK messages:

$$\begin{aligned} &ack(31), ack(32), nak(33), nak(34), \\ &nak(35), ack(36), ack(37), ack(38) \end{aligned} \quad (4)$$

Such sequence could be split into the several subsequences containing only ACK messages:

$$AckSeq_i = \{ack_0(N_i), \ldots, ack_{n_i}(N_i + n_i)\} \quad (5)$$

and NAK messages:

$$NakSeq_i = \{nak_0(N_i), \ldots, nak_{n_i}(N_i + n_i)\} \quad (6)$$

where n_i+1 is the number of ACK or NAK messages in each subsequence.

Because numbers of received payload packets are constantly increasing, then original sequence could be compressed to the following sequence:

$$ack(31), nak(33), ack(36), ack(38) \quad (7)$$

in general *m* subsequences could be compressed to the following sequence:

$$ack_0(N_0), nak_0(N_1), ack_0(N_2), \ldots \\ \ldots, ack_0(N_{m-1}), ack_{n_{m-1}}(N_{m-1} + n_{m-1}) \quad (8)$$

It is necessary to send an empty payload packet every 2 seconds to the receiver, when there is no payload data to send. It is used for computing current RTT. Empty payload packets also work as a keep alive packets. When the host does not receive any packet from its peer during 30 seconds, then this connection is considered as closed.

Tests of Verse protocol

The Verse protocol was tested in similar client-server application where transport protocols were tested. The Verse server run again on virtualized OS. Position of moving particles were sent to the Verse server from special Verse client running on the same virtualized OS. When the Verse server received positions of particles, then it tried to send these positions to the second Verse client running at host OS. The link between host and virtualized OS was modified in the same way as it is described in section 2. We can see at Fig. 8 that average delay of Verse protocol was comparable with average delay of UDP. The congestion was longer and delay was bigger than with UDP, because messages containing position of particles are not so simple as messages used for tests of transport protocols. It is price for flexibility and partial reliability of Verse protocol.

6. RELATED WORK

Work of Terrence L. Disz et. al. [DPPS95] contains first experiments with CAVE to CAVE communication. They proposed very ambitious plans of object sharing, but it was only plan and this project was canceled. Chen-Chi Chet et. al. [WCC+ 09] proposed Game Transport Protocol (GTP) with 4 schemes of re-transmission, but every type of retransmission scheme can re-transmit whole packet. This approach is very similar to SCTP protocol and it's inefficiency for ASVR was proved. Harcsik et. al. [HPG07] tested transport protocol and its efficiency for network games. These test were specific for network games using thin streams – they consists of small packets sent at low packets rates.

7. CONCLUSION AND FUTURE WORK

Sharing of 3D data over lossy networks is quite challenging problem. UDP, TCP, SCTP and DCCP transport protocols were tested and compared in special client-server application. It was proved that no transport protocol as is can guarantee low delays together with reliable or semi-reliable transport. Basic principles of new Verse protocol were introduced. This protocol was also tested in the client-server application. Proposed approach used in Verse protocol gave significantly better results for ASVR than simple ad-hoc solutions based on transport protocols, because Verse resend mechanism re-sends only actual data and obsolete data are dropped. New Verse protocol allows to use compression and thus further minimize congestion and delays.

Future work will be focused on design and implementation of reliable congestion control for datagram transport varying packet size with fixed sending rate of packets. Next target will be implementation of prioritization, queueing and scheduling of data, that are going to be sent to the receiver. In this way it will be able to increase probability of delivering data with high priority.

8. ACKNOWLEDGMENTS

This research is realized under the state subsidy of the Czech Republic within the research and development "Advanced Remediation Technologies and Processes Center" 1M0554 – Program of Research Centers PP2-DP01 supported by Ministry of Education. I would like to thanks to Pavel Satrapa and David Kmoch for useful suggestions and careful reading of this paper.

9. REFERENCES

[ARA04] Al-Regib, G. Altunbasak, Y. 3TP: 3-D models transport protocol. In Web3D '04: Proceedings of the ninth international conference on 3D Web technology, pages 155–162, New York, NY, USA, 2004. ACM.

[DPPS95] Disz, T.L. Papka, M.E. Pellegrino, M. Stevens, R. Sharing Visualization Experiences among Remote Virtual Environments. In International Workshop on High Performance Computing for Computer Graphics and Visualization, pages 217–237. Springer-Verlag, 1995.

[KHF06] Kohler, E. Handley, M. Floyd, S. Designing DCCP: congestion control without reliability. In SIGCOMM '06: Proceedings of the 2006 conference on Applications, technologies, architectures, and protocols for computer communications, pages 27–38, New York, NY, USA, 2006. ACM.

[KCJ07] Kempf, J., Chander, A., and Jo, M. Optimizing avatar environmental update in shared virtual reality environments. In Proceedings of the First International Conference on Immersive Telecommunications (ICST, Brussels, Belgium, Belgium, 2007), ImmersCom '07, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–6.

[Hem05] Hemminger, S. Network Emulation with NetEm. In Linux Conf Au, April 2005.

[HPG07] Harcsik, S., Petlund, A., Griwodz, C., and Halvorsen, P. Latency evaluation of networking mechanisms for game traffic. In Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games (New York, NY, USA, 2007), NetGames '07, ACM, pp. 129–134.

[Pos80] Postel, J. RFC 768: User Datagram Protocol, aug 1980.

[Pos81] Postel, J. RFC 793: Transmission Control Protocol, sep 1981. Updated by RFCs 1122, 3168.

[BSS06] Brink, E. Steenberg, E. Svenson, G. The Verse Networked 3D Graphics Platform. In SIGRAD '06: Conference proceedings: The Annual SIGRAD conference: Special theme: Computer Games, pages44-48, Skövde, Sweden. 2006.

[SB07] Steenberg, E. Brink, E. The Verse Specification. http://verse.blender.org/, 2007.

[Ste07] Stewart, R. RFC 4960: Stream Control Transmission Protocol, sep 2007.

[WCC+09] Wu, C.C. Chen, K.T. Chen, C.M. Huang, P. and Lei, C.L. On the challenge and design of transport protocols for MMORPGs. Multimedia Tools Appl., 45(1-3):7–32, 2009.

[LG09] Linux Foundation. Networking ToDo List. http://www.linuxfoundation.org/collaborate/workgroups/networking/todo, 2009

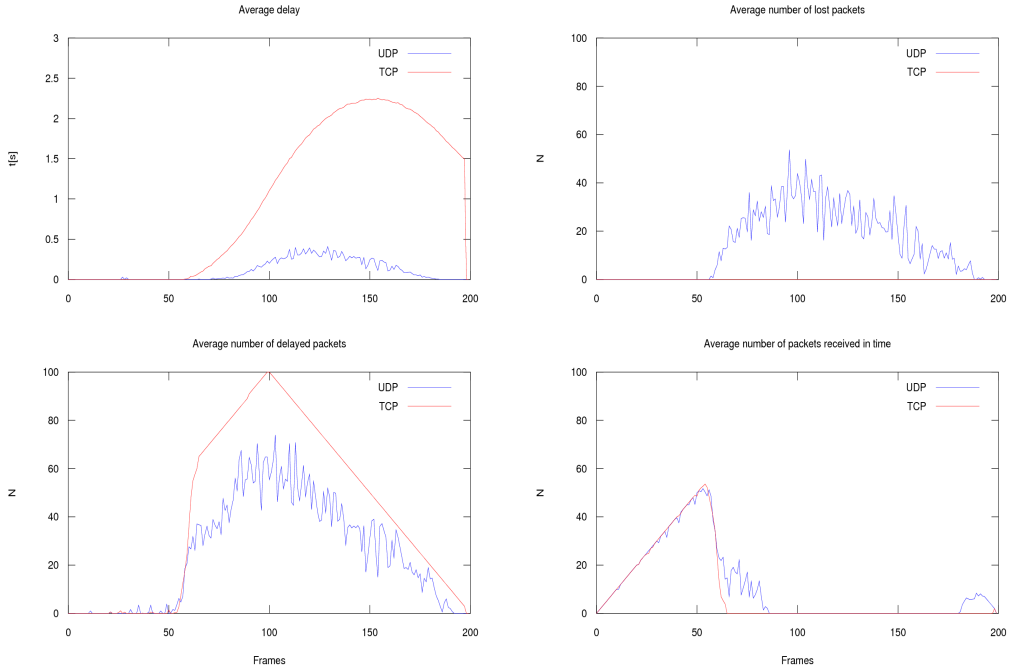


Figure 4: Results of experiments with 1000 particles on real WAN network. The link had delay about 5 ms (delay jitter 1 ms) and the bandwidth was about 1900 kb/s.

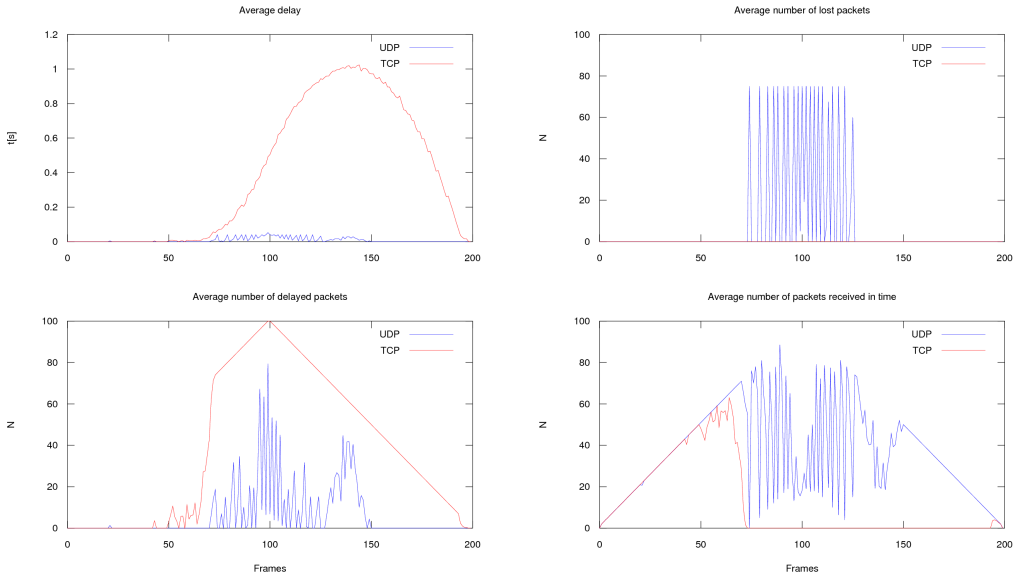


Figure 5*:* Test of UDP protocol proofed, that UDP had average delay quite low. Delay between two frames was 40 milliseconds (~25 FPS). Some of particles lost after 100 th frame has never been resend and this packet loss caused inconsistency of data between client and server.

Test of TCP protocol proofed, that this transport protocol is not feasible for ASVR. When generated traffic exceeded bandwidth, then delay of received particles grooved to 1 second.

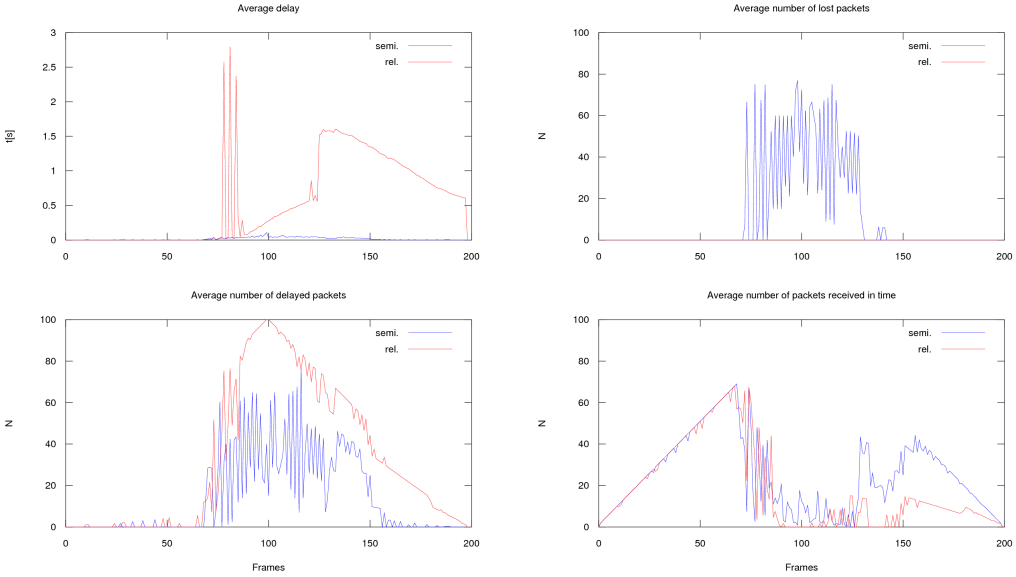


Figure 6: Test of semi-reliable variant of SCTP protocol gave similar results as UDP protocol. Average delay was quite low, but lost packets were not resend and it caused inconsistency in data between server and client, when transmission of particles was finished.

Test of reliable variant of SCTP protocol. This protocol gave similar results as TCP protocol. When generated traffic exceeded bandwidth of the link between client and server, then delay grooved up to 3 seconds.

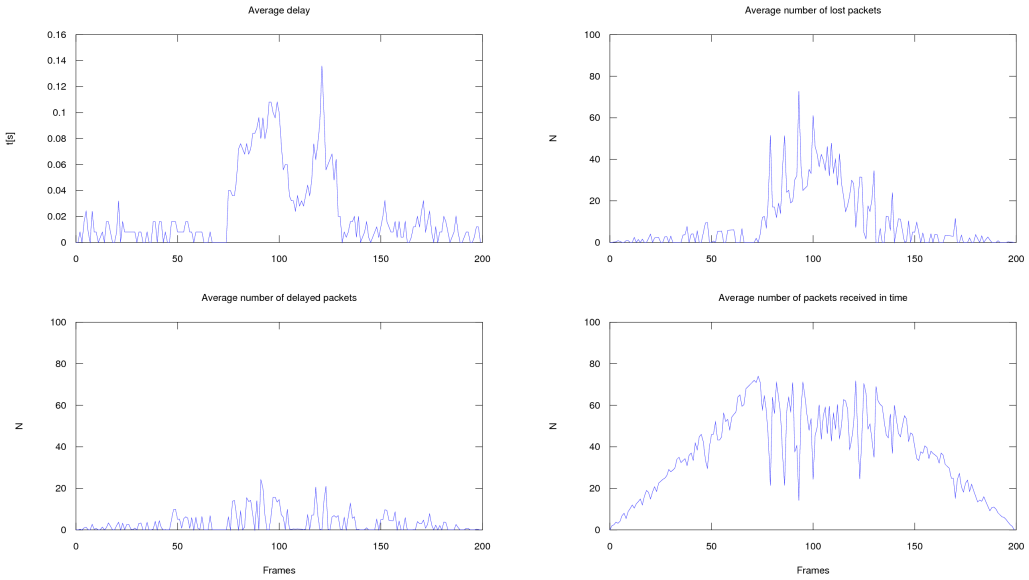


Figure 7: Test of Verse protocol has bigger average delay, then UDP or semi-reliable variant of SCTP protocol, because Verse protocol has to transfer more data (header, acknowledgment commands, node commands, etc.). When transferring of particles was finished, then position of all particles was the same at the client and the server.

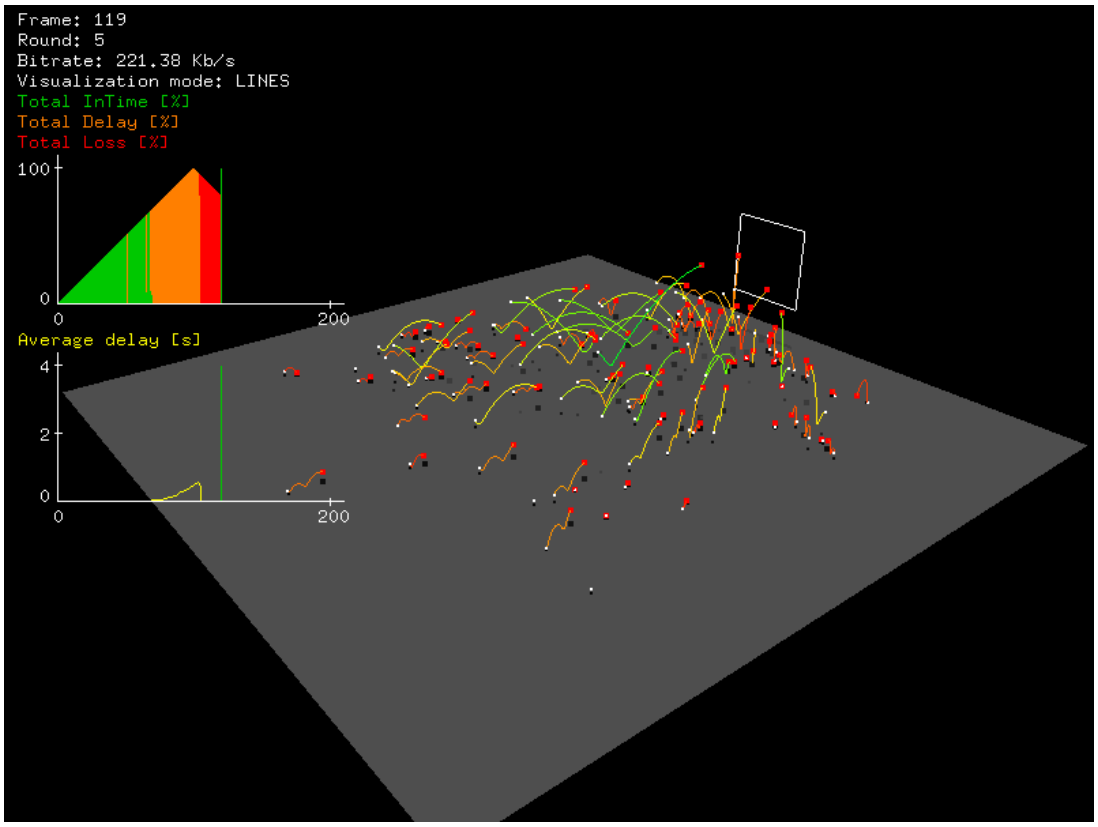


Figure 8: Screenshot of client visualizing delay of received particles. TCP protocol is used in this case. This screenshot was captured during congestion. Thus all received particles are delayed. Red points visualize current received positions and white points visualize expected positions of particles. Colored line between these two points visualize delay of received particle.

RaVioli: a GPU Supported High-Level Pseudo Real-time Video Processing Library

Katsuhiko KONDO* Takafumi INABA* Hiroko SAKURAI†

Masaomi OHNO* Tomoaki TSUMURA* Hiroshi MATSUO*

* Nagoya Institute of Technology, Japan
camp@matlab.nitech.ac.jp

† OMRON Corp., Japan.

ABSTRACT

Real-time video processing applications such as intruder detection system are now in demand and being developed. However, on general purpose computers, it is difficult to guarantee that enough CPU resources can be surely be provided. We have proposed a pseudo real-time video processing library RaVioli for solving this problem. RaVioli conceals two types of resolutions, frame rate and the number of pixels, from programmers. This makes video and image processing programmings more intuitive, but the performance may be lower by the abstraction overhead. To solve this problem, this paper proposes an improvement of RaVioli for supporting GPU platforms. For using GPUs effectively, a deep knowledge about them has been required, and this would have been a burden to programmers. The proposition on this paper provides an easy-to-use framework for developers. They can benefit from GPU without rewriting their RaVioli programs and get high performance video processing. The experiment results with image/video processing programs show that the proposed method improves the performance about 151-fold/164-fold in maximum against traditional RaVioli without rewriting programs, and about 30-fold/4-fold in maximum against a native C++ program.

Keywords

real-time video processing, programming paradigm, video processing library, CUDA

1 INTRODUCTION

The demand of the systems, which highly requires real-time video processing, is rapidly increasing; such as intruder detection systems, automatic vehicle collision avoidance systems, and so on. It is also expected that the performance improvement and the cost reduction will promote real-time video processing on the general-purpose computers and operating systems. In spite of the advances, it is difficult to realize the real-time video processing on general-purpose operating systems, because it should be run by constant time interval. The main reason of the difficulty is the fluctuation in the throughput of frame rate and in the amount of the available CPU resources.

To solve this problem, we have proposed a high-level video processing library *RaVioli* (Resolution-Adaptable Video and Image Operating Library) which guarantees pseudo real-time processing on general-purpose system platforms. RaVioli can regulate the throughput rate by automatically fluctuating spacial resolution and frame rate according to CPU usage and load. For such dynamical fluctuation of the resolutions, a programming fashion which is independent of the resolutions is required. RaVioli conceals two resolutions, the spacial resolution of an image and the frame rate of a video stream, from programmers for controlling the resolutions automatically at run-time. It makes possible to exclude the concept of resolutions, and developers can write video processing programs more intuitively.

However, RaVioli causes the decline of processing speed that comes from the abstraction overhead. Hence, to solve this issue, this paper proposes an improvement of RaVioli to support CUDA GPU platforms which are ideally suited to multimedia processing. The proposition of this paper is the method to provide an easy-to-use programming framework. Developers can implement real-time video processing programs without considering GPU architectures, and achieve high performance video processing.

2 RESEARCH BACKGROUNDS

2.1 Related Works

For real-time video processor, adjusting the processing load is very important. Nevertheless, writing multiple routines with different algorithms has been the only solution for the load adjustment. One example

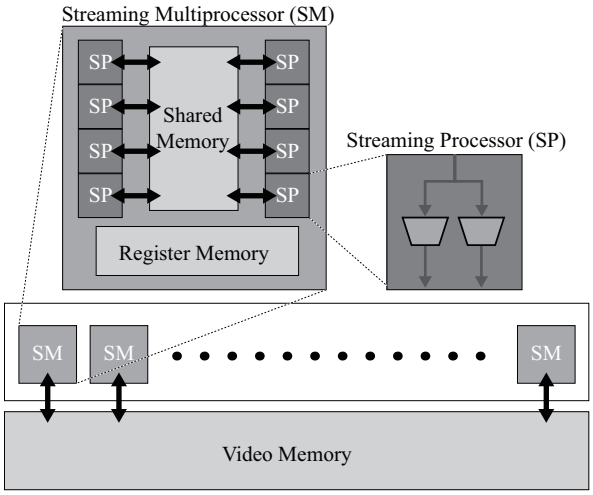


Figure 1: Brief architecture of GPU

that has been proposed is the imprecise computation model (ICM)[1, 2]. In this model, computation accuracy is varied corresponding to the given computation time. With the confidence-driven architecture, which is based on the ICM, developers have to troublesomely implement multiple routines with different algorithms and different loads, and the confidence-driven architecture selects suitable routine dynamically and empirically among them.

VIGRA[3] and OpenCV[4, 5] are well-known video processing libraries. They aim at high-level descriptivity of video processing. Adopting template techniques similar to the C++ STL, VIGRA allows programmers to easily adapt given components to their programs. OpenCV provides many typical video processing algorithms as C functions or C++ methods. However, adjusting computation load is difficult to be implemented with these libraries.

The approach of a library RaVioli[6] is completely different from these existing computation models or video processing libraries. RaVioli allows programmers to be unaware of the existence of pixels and frames through their video processing programming. Concealing pixels and frames from programmers, RaVioli can vary spacial/temporal resolutions and can adjust processing load dynamically and automatically.

2.2 GPU and CUDA

A *GPU (Graphics Processing Unit)* is a specialized microprocessor for image/video processing. It has wide memory bandwidth and high processing performance. A brief architecture of a GPU shipped by NVIDIA Corp. is shown in Figure 1. Although GPUs are different according to their generations and families, they have a common architecture. There are dozens of *Streaming Multiprocessors (SM)* in one GPU, and for example in the GT200 series, each SM has eight *Streaming Processors (SP)*. The eight SPs in an SM can run in SIMD (Single Instruction Multiple Data) fashion. A GPU is a massively parallel multi-core processor, for example, a GT200 series GPU has 30 SMs, and consequently has 240 SPs.

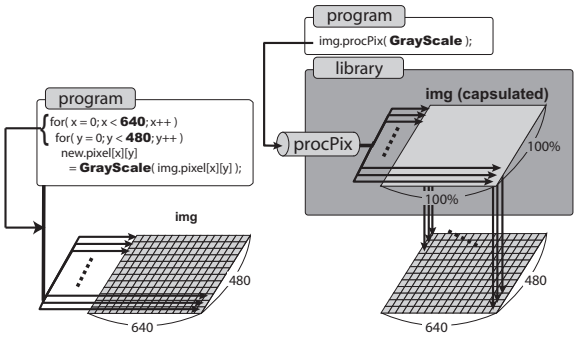
NVIDIA also provides *CUDA (Compute Unified Device Architecture)*[7] for GPU programming. CUDA is a parallel computing architecture for GPUs. CUDA also includes compilers and libraries, and provides APIs for GPU programming. Hence, developers can easily access memories of the computational units in GPUs using CUDA. GPUs can achieve high performance by executing massively parallel threads simultaneously. In the CUDA framework, a GPU can execute $65535 \times 65535 \times 512$ threads across all SPs. CUDA organizes these threads into two levels of units; *Grid* and *Block*. A Block is executed on an SM, and the threads in a Block can be identified by three-dimension indices (x, y, z). A set of Blocks is called Grid, and the Blocks in a Grid can be identified by two-dimension indices. How many threads are associated to a Block and how many Blocks per Grid are called as an *execution configuration*. Defining an appropriate execution configuration is a key for achieving good performance on GPU, but it is rather difficult for ordinary programmers.

Hence, some frameworks are proposed for CUDA programming. Baskaran et.al.[8] has proposed a translator for optimizing CUDA programs. It makes global memory accesses effective. The compiler framework optimizes affine loop nests based on a polyhedral compiler model. CUDA-lite[9] is another translator for CUDA programs. It generates a optimized code which uses appropriate GPU memories. Lee et.al.[10] has also proposed a optimization framework for GPU programs. However when using these frameworks, developers should pay attention to parallelism, and should add annotations or pragmas for getting efficient code. On the other hand, since RaVioli hides loop iterations from developers, essential parallelism or data dependencies between iterations are easily found automatically.

3 OVERVIEW OF RAVIOLI

3.1 Abstraction of Video Processing

RaVioli[6] proposes a new programming paradigm with which programmers can write video processing applications intuitively. RaVioli conceals *spatial resolution* (pixel rate) and *temporal resolution* (frame rate) of a video from programmers. We human beings naturally have no concept of resolutions through our visual recognition. For example, we can recognize object motion in our view without any pixel or frame. However, pixels and frames are indispensable



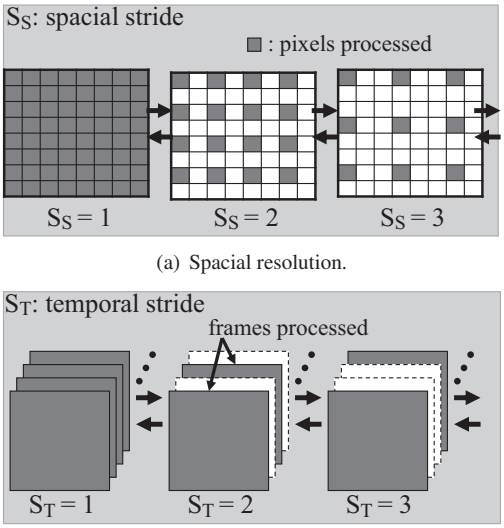
(a) Traditional program. (b) Program with RaVioli.

Figure 2: Digital image processing.

for motion object detection programs on computer systems.

For example, motion object detection programs are sometimes implemented by using a block matching algorithm, which searches the most similar block between current window and previous one. The similarity between image windows will be calculated by SAD (sum of absolute differences) or other alternative methods, and the methods should be implemented by cumulative pixel value differences. Resolutions are delivered from the requirement of quantitativeness on computers. Hence, programmers have to manage resolutions in their programs although resolutions are not required essentially for vision. In other words, the presence of resolutions makes programs unintuitive.

Generally, loop iterations are heavily used in video processing programs. When converting a color image to grayscale, for example, each pixel will be converted to grayscale in innermost iteration, and the process is repeated for every pixels by loop nests as shown in Figure 2(a). In RaVioli, an image is encapsulated in an RV_Image instance, and this repeating process for all pixels is done by RaVioli automatically, so programmers should only write a routine for one pixel as shown in Figure 2(b). GrayScale() in Figure 2(b) is the routine defined by the programmer. What programmer should do are defining function which processes one pixel and passing the function to an image instance's public method procPix(). The proxPix() is defined as a higher-order method which applies a function passed as its argument to all pixels one after another. This framework allows programmers to be released from resolutions and the number of iterations. Not only procPix(), RaVioli also provides some higher-order methods for several processing patterns; such as template matching, k-neighbor processing, and so on. As same as images, videos are also encapsulated in RV_Video instances in RaVioli. Frames, the components of an RV_Video instance, are concealed from developers. An RV_Video instance also has several higher-order methods. Developers should only define a component function, which



(a) Spacial resolution.

(b) Temporal resolution.

Figure 3: Resolution changes.

manages one frame, and pass the function to an appropriate higher-order method for video processing.

Pseudo real-time processing and parallelization are also resolved by RaVioli. RaVioli conceals resolutions from programmers, therefore RaVioli can easily vary resolutions through real-time processing for load reduction. Moreover, the iteration unit is so distinct in RaVioli programs that the programs can be automatically data-parallelized. Sakurai et.al. have taken these functions up in detail in [6].

3.2 Self-Adjustment of Computation Load

RaVioli can dynamically vary video resolutions considering processing load. RaVioli periodically compares the frame capturing interval and the processing time for one frame. When the processing time becomes larger than the capture interval, RaVioli considers it is overloaded and reduces resolutions. There are two resolutions; spatial resolution and temporal resolution in videos. Spatial resolution refers the number of pixels contained in each frame, and temporal resolution refers the frame rate. RaVioli applies component functions to frames or pixels skipping on a certain stride in higher-order methods mentioned above. Roughening resolutions can be done by raising the stride value, and it leads to decreasing the computation load. Figure 3(a) shows which pixels are processed when special stride increases, and Figure 3(b) shows which frames are processed when temporal stride increases.

Priorities can be specified for telling RaVioli which resolution (special or temporal) should be kept. In a real-time video application, top priority will be given to temporal resolution, and RaVioli reduces spatial resolution. In other applications such as face authentica-

tion, top priority will be given to spatial resolution, and RaVioli reduces temporal one. What should be done for load adjustment is only specifying priorities.

The resolution priority is specified by a tuple of two values (P_S, P_T) called a *priority set*. P_S represents the priority of spatial resolution, and P_T the priority of temporal resolution. When (P_S, P_T) = (3, 7) is specified, the priority ratio of P_S and P_T is recognized as 3:7, and RaVioli manages to keep spatial stride and temporal stride in the ratio of 7:3. Therefore a video processing application, which fulfills the performance demand and realizes real-time processing, can be easily implemented.

This algorithm for reducing resolutions is very simple and naive. However, this simplicity is very important. Many complement algorithms such as bi-linear, hypercubic, and so on are well known and they can be used. However, notice that the function of changing resolutions of RaVioli aims at reducing calculations. Adding calculations for changing resolutions makes no sense. An application written with RaVioli can achieve real-time processing without any considerations. Sometimes the output will have low quality, but the application does not lose *realtimeness*. Moreover, defining priority set appropriately can control the inconvenience from the quality loss.

4 CUDA SUPPORT FOR RAVIOLI

In this section, CUDA-supported RaVioli (RaVioli/CUDA) and a translator which converts traditional RaVioli programs to programs for RaVioli/CUDA are proposed.

4.1 Execution Model of Image Processing with CUDA

Developers can use several memories of GPU with CUDA. Each memory has different access speed and size. For achieving high performance image processing, developers should use as fast memory as possible. For using fast memories, data should be transferred from main memory to GPU memories. Considering these memories and execution configurations needs deep knowledge and dexterity.

In this paper, we propose an extension of higher-order methods of RaVioli which supports CUDA API. Invoking these higher-order methods, developers can use GPUs without considering GPU memories, execution configurations and other troublesome steps. Figure 4 briefly shows how a GrayScale() function will be applied to an image by invoking the extended higher-order method cudaProcPix(). GrayScale() which is to be passed to cudaProcPix() should be defined as a *kernel function*. In CUDA, a kernel function specifies the code to be executed by all threads in parallel.[7]

When cudaProcPix() is invoked with a component function GrayScale(), RaVioli/CUDA allocates GPU

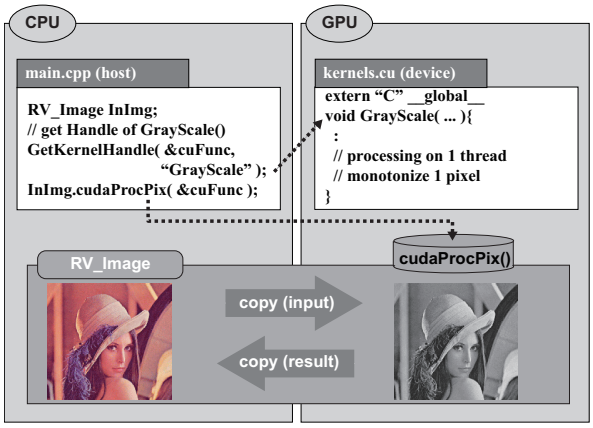


Figure 4: Brief execution model of RaVioli/CUDA

memories and transfers image data from main memory to GPU memories. After that, an execution configuration is automatically defined according to the input image, and GrayScale() is applied to the whole input image. When completing the application, RaVioli/CUDA transfers the result to the main memory on CPU and deallocates GPU memories. For each other higher-order method of RaVioli, an associated CUDA-supported method is defined.

4.2 Execution Model of Video Processing with CUDA

As same as RV_Image class, CUDA-supported higher-order methods for RV_Video class are also defined. The methods for RV_Video not only conceals data transfer between CPU and GPU, but also parallelize the data transfer and kernel function execution automatically by using *CUDA stream*.

In CUDA, the execution of a kernel function and the data transfer between host (CPU) and device (GPU) can be overlapped by using multiple CUDA streams. A CUDA stream is defined as a sequence of CUDA operations which are executed in-order. Multiple CUDA streams can be declared and used simultaneously. Each data transfer between host and device and each execution of kernel function can be assigned to one of the defined CUDA streams. A host-device data transfer and a kernel function execution on different CUDA streams can be executed in parallel.

In RaVioli/CUDA, two CUDA streams are automatically declared when an RV_Video is instantiated. When a higher-order method of the RV_Video instance is invoked, each frame of the video is assigned to the two CUDA streams alternately. The execution model is illustrated in Figure 5.

First, the stream #1 transfer the frame #1 from host to device. When the transfer completes, the stream #2 can transfer the frame #2, and the stream #1 applies kernel

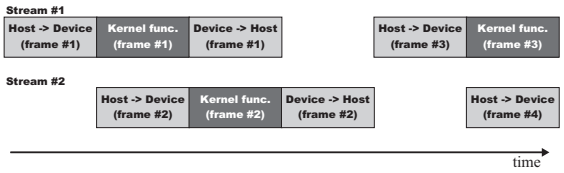


Figure 5: Pipelining with CUDA streams.

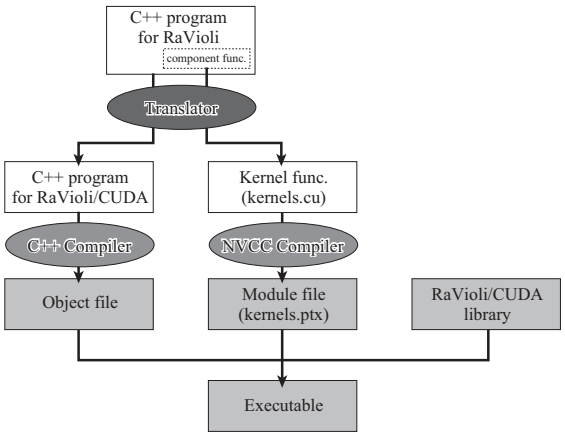


Figure 6: Compilation flow with translator.

function to the frame #1 simultaneously. Notice that the stream #2 cannot transfer the frame #3 as soon as the result of the frame #1 is sent back to the host, because the stream #2 will be send the result of the frame #2. This brings a pipeline bubble. However, in the ideal case in which every stage takes same latency, the throughput raises 1.5-fold.

4.3 Translator and Code Conversion

As described above, RaVioli/CUDA can provide an easy-to-use CUDA programming framework for developers. Higher-order methods of RaVioli/CUDA conceal almost all steps for using CUDA such as device handling, memory allocation, execution configuration, and so on from developers. However, developers still should modify their traditional RaVioli programs. For example in Figure 4, the developer should get a handler for a kernel function by calling GetKernelHandler(), and rewrite the higher-order method procPix() to the associated CUDA-supported higher-order function cudaProcPix(). The component functions also should be rewritten to kernel functions for being adapted to CUDA-supported higher order methods. To dissolve these troubles, a translator which converts traditional RaVioli programs to RaVioli/CUDA programs is also proposed in this paper. Figure 6 shows a compilation flow with the translator.

Figure 7 shows an example program which converts color images to grayscale. The translator converts such programs to two program files; `main.cpp` for host CPU and `kernels.cu` for GPU device. These program files are compiled by C++ compiler and CUDA compiler `nvcc`, and assembled to an executable. The result of conversion is shown in Figure 8 and Figure 9.

```
1  int main(int argc, char* argv[]){
2  RV_Image image;
3      :
4    image.procPix( GrayScale );
5      :
6  }
7
8  void GrayScale(RV_Pixel p1){
9    int ave = ( (p1.getR()) + p1.getG() + p1.getB() ) / 3;
10   p1.setRGB( ave, ave, ave );
11 }
```

Figure 7: A simple grayscale program with traditional RaVioli.

```
1  /* main.cpp */
2  RV_CudaDevice device;
3  int main(int argc, char* argv[]){
4    RV_Image image;
5    :
6    device.RaCudaInit(); /* initialize device */
7    CUfunction cuFunction;
8    device.GetKernelHundle( &cuFunction, "GrayScale" );
9    image.cudaProcPix( &cuFunction );
10   :
11   device.RaCudaExit(); /* finalize device */
12 }
```

Figure 8: Main program translated from Figure 7.

```
1  /* kernels.cu */
2  extern "C" __global__ void
3  GrayScale( RV_Pixel* idata, RV_Pixel* odata, int width, int height ){
4    int x = blockDim.x * blockIdx.x + threadIdx.x;
5    int y = blockDim.y * blockIdx.y + threadIdx.y;
6    RV_Pixel p1;
7    if( x < width && y < height ){
8      p1 = idata[ y * width + x ];
9      int ave = ( p1.getR() + p1.getG() + p1.getB() ) / 3;
10     odata[y * wid + x].setRGB( ave, ave, ave );
11   }
12 }
```

Figure 9: Kernel program translated from Figure 7.

In the main program, the invocation of procPix() in Figure 7 is converted to cudaProcPix() in Figure 8. A statement of GetKernelHandler() is added in main() for getting a kernel handler for the component function GrayScale(). RaCudaInit() and RaCudaExit() are functions provided by RaVioli/CUDA for CUDA device initialization and finalization respectively.

On the other hand in the kernel program, the component function GrayScale() is converted to a kernel function. A kernel function expresses a process for one thread. In Figure 9, the kernel function GrayScale() is defined as it processes one pixel on one thread. The definition of GrayScale() also makes continuous threads to process continuous pixels by calculating indices. This

is for coalesced access of CUDA memories. Memory accesses to global memory by continuous sixteen threads in each *Block* can be issued in parallel by this code conversion.

The translator searches higher-order method invocations through RaVioli programs, and generates associated code for RaVioli/CUDA with converting component functions to kernel functions. In this simple example program, there need no reduction operation for parallelization. However, in RaVioli programs, whether reduction operations are required or not can be easily detected, because any dependency between iterations appears as an assignment to a *global variable* in the component function.[6]

Enumerating translation rules in detail is left out for want of space. There are additional functions of the translator as follows.

Optimizing Data Transfers Many of video processing programs consist of multiple stages, and the stages can be pipelined. Since transferring data between CPU and GPU on each stage of the pipeline is redundant, the translator optimizes these data transfers. In the output program converted by the translator, the data are transferred from CPU to GPU only once at the first stage (i.e. the first invocation of a higher-order method), and the result is transferred from GPU to CPU only once at the last stage.

Using Page-Locked Memory With CUDA, two types of CPU host memory are available. The one is heap memory, and the other is page-locked host memory. Page-locked host memory is mapped into the address space of the GPU device, and can be accessed directly. Moreover, data copy between page-locked host memory and GPU can be fast and asynchronous. The translator converts programs for using page-locked memory automatically.

5 EVALUATION RESULTS

A CUDA extension for RaVioli and a translator described in section 4 were implemented, and evaluated with several image/video processing programs. The evaluation environment is shown in Table 1.

5.1 Evaluation of Image Processing

We used three programs which are grayscale, emboss filter and template matching for evaluating image processing. The evaluation results are shown in Table 2. In Table 2, *Baseline* denotes a program written in native C++, *RaVioli* denotes a program with traditional RaVioli and *RaVioli/CUDA* denotes a program with CUDA-supported RaVioli described in this manuscript. The size of the image which was used for grayscale and emboss filter was 512×512 pixels. For template matching, the base image has 395×372 pixels and the template image has 70×72 pixels.

OS	Fedora9
CPU	Core2Quad
Frequency	2.83GHz
Memory	3GB
GPU	GeForce GTX280
Number of multiprocessors	30
Number of cores (SP)	240
CUDA version	2.2 (Driver API)
Compute capability	1.3
Compiler	gcc
Compile options	-O3

Table 1: Evaluation environment.

Workloads	*Baseline*	*RaVioli*	*RaVioli/ CUDA*
Grayscale	0.83	2.89	1.21
Emboss filter	2.08	18.62	1.30
Templ. matching	1902.45	9512.69	62.62

Table 2: Execution time. (ms)

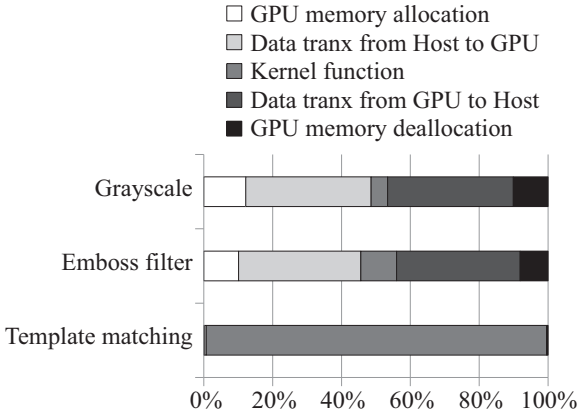


Figure 10: Breakdown of processing time with RaVioli/CUDA

As we can see in Table 2, *RaVioli/CUDA* achieves performance gains of 2.3-fold, 14.2-fold and 151.8-fold on grayscale, emboss filter and template matching respectively, against traditional RaVioli without rewriting programs. Typically on template matching, *RaVioli/CUDA* also achieves about 30-fold speedup against *Baseline*.

Nevertheless, the performance of RaVioli/CUDA on grayscale is still inferior to *Baseline*. Hence, the breakdown of processing time with *RaVioli/CUDA* was also evaluated. The result is shown in Figure 10. As we can see in Figure 10, the kernel function execution dominates the whole execution time on the template matching program, and the parallelization on GPU brings a

good result. On the other hand, the kernel function execution accounts for about only 4% of total execution on the grayscale program, and the data transfer overheads between host CPU and GPU device are dominant. This should prevent performance gain on the grayscale program. However, a program with small kernel function does not bring large latency and will not cause a problem on real-time video processing essentially.

5.2 Evaluation of Video Processing

The performance of video processing with RaVioli/CUDA was also evaluated. We have evaluated four models with an edge detection program. Roughly speaking, the edge detection processing consists of three stages; converts input frames to grayscale, binarize it, and detects object edges. The results of processing time for ten frames are shown in Figure 11.

RaVioli/CUDA achieved about 90-fold speedup against traditional RaVioli without rewriting program, and over 2-fold speedup against the baseline program. Furthermore, RaVioli/CUDA with CUDA stream achieved 164-fold speedup against traditional RaVioli, and about 4-fold speedup against the baseline. As we can see in the breakdown of the third bar, the latency of data transfer is longer than the latency of kernel function execution. However, the result overcomes the ideal 1.5-fold speedup mentioned in Chap. 4.2. This can be explained by additional functions of the translator mentioned in Chap. 4.3.

As a result, RaVioli/CUDA provides a high-level programming framework for developers. Developers can use GPU without any knowledge and consideration, and can achieve high performance by using RaVioli/CUDA. On expensive programs, GPU abilities can be easily brought out by RaVioli/CUDA, and on lightweight programs, RaVioli/CUDA can limit the abstraction overhead of RaVioli effectively.

6 CONCLUSIONS

In this paper, we have proposed an improvement of RaVioli for supporting CUDA GPU platforms. RaVioli is a pseudo real-time video processing library which conceals spacial/temporal resolutions from programmers and changes resolutions automatically for adapting to currently available CPU resource. RaVioli/CUDA not only allows developers to be free from considering GPU architectures, but also easily brings out the performance in GPU devices.

The evaluations with several image and video processing programs have been conducted. The results with image processing programs have shown that RaVioli/CUDA achieves 151-fold speedup in maximum against traditional RaVioli without rewriting programs, and also achieves about 30-fold speedup against native C++ programs. The results with a video processing program of edge detection has shown

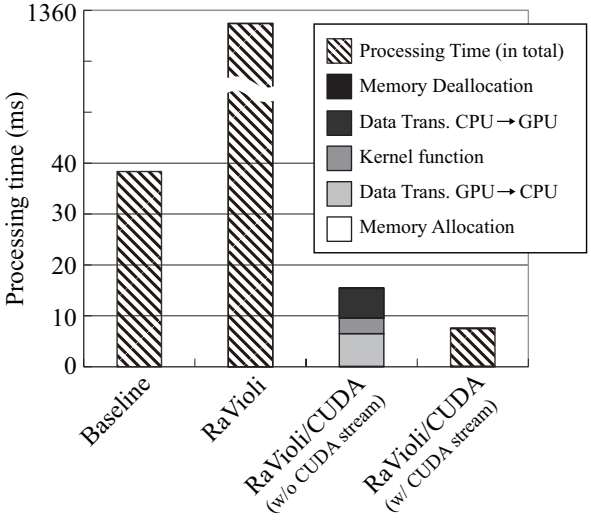


Figure 11: Evaluation of Video Processing.

that RaVioi/CUDA achieves about 164-fold speedup against traditional RaVioli and about 4-fold speedup against a native C++ program.

Possible improvement of this study is modifying execution configuration appropriately and dynamically. This can make kernel functions to run more effectively. Now, RaVioli has good writeability, and many programs such as edge detection, circle detection, hough transform, and so on can be written with RaVioli. However, image reconstruction and frequency processing are hard to be written with RaVioli at the moment. We should examine some new higher-order methods for them. Designing a new video programming language which cooperates with RaVioli is also left for our future work.

ACKNOWLEDGEMENTS

This research was partially supported by a Grant-in-Aid for Young Scientists (B), #21700028, 2009, from the Ministry of Education, Science, Sports and Culture of Japan.

REFERENCES

[1] J.W.S. Liu, Wei-Kuan Shih, Kwei-Jay Lin, R. Bettati, and Jen-Yao Chung. Imprecise Computations. In *Proceedings of the IEEE*, volume 82, pages 83–94, Jan. 1994.

[2] Hiromasa Yoshimoto, Naoto Date, Daisaku Arita, and Rinichiro Taniguchi. Confidence-Driven Architecture for Real-time Vision Processing and Its Application to Efficient Vision-based Human Motion Sensing. In *Proc. of the 17th Int'l. Conf. on Pattern Recognition (ICPR'04)*, volume 1, pages 736–740, 2004.

[3] Ullrich Köthe. *VIGRA - Vision with Generic Algorithms*, 1.6.0 edition, Aug. 2008.

[4] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer Vision With the OpenCV Library*. O'Reilly & Associates Inc, 2008.

[5] Intel Corp. *Open Source Computer Vision Library*, 2001.

[6] Hiroko Sakurai, Masaomi Ohno, Tomoaki Tsumura, and Hiroshi Matsuo. RaVioli: a Parallel Video Processing Library with Auto Resolution Adjustability. In *Proc. IADIS Int'l. Conf. Applied Computing 2009*, volume 1, pages 321–329, Nov. 2009.

[7] NVIDIA Corp. *NVIDIA CUDA Programming Guide*, 2.0 edition, Jun. 2008.

[8] Muthu Manikandan Baskaran, Uday Bondhugula, Sriam Krishnamoorthy, Atanas Rountev, and P.Sadayappan. A Compiler Framework for Optimization of Affine Loop Nests for GPGPUs. In *ICS'08: Proc. of 22nd Annual Intl. Conf. on Supercomputing*, pages 225–234. ACM, 2008.

[9] Sain-Zee Ueng, Melvin Lathara, Sara Baghsorkhi, and Wen mei Hwu. CUDA-lite: Reducing GPU Programming Complexity. In *Proc. of 21st Annual Workshop on Languages and Compilers for Parallel Computing (LCPC 2008)*, pages 1–15, 2008.

[10] Seyong Lee, Seung-Jai Min, and Rudolf Eigenmann. OpenMP to GPGPU: A Compiler Framework for Automatic Translation and Optimization. In *Proc. of the 14th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, volume 44, pages 101–110. ACM, 2009.

APPENDIX

Another example of code conversion by the translator is shown in this appendix. Figure 12 is a template matching program written with traditional RaVioli. The codes shown in Figure 13, Figure 14 and Figure 15 are generated by the translator from the code in Figure 12.

```
1  RV_Image tp_image;
2  RV_Coord start;
3  RV_Coord end;
4  int sad;
5
6  void SAD(RV_Pixel p1, RV_Pixel p2){
7    int abs = p1.absDiff(p2);
8    sad += abs;
9  }
10
11 void TPmatching(RV_Image imageSmall,
12                   RV_Coord startNow, RV_Coord endNow){
13   sad = 0;
14   int min = INT_MAX;
15   imageSmall.procImgComp(SAD, tp_image);
16   if(min > sad){
17     min = sad;
18     start = startNow;
19     end = endNow;
20   }
21 }
22
23 int main(int argc,char* argv[]){
24   RV_Image* input_image = new RV_Image(argv[1]);
25   RV_Image* output_image = new RV_Image(argv[1]);
26   tp_image = new RV_Image(argv[2]);
27   input_image->procBox(TPmatching,
28                        input_tp->getStartCoord(),
29                        input_tp->getEndCoord());
30   output_image->writeRect(start,end);
31   return 0;
32 }
```

Figure 12: A template matching program with traditional RaVioli.

The component function SAD() is converted to the SAD() in Figure 14, and the component function TP-

```
1  /* main.cpp */
2
3  #include "ravioli.h"
4  #include "cutil.h"
5
6  RV_Cuda device;
7
8  CUtexref cuTexTPref; // texture reference for template image
9  CUarray d_TPimage;
10 int3 result;
11
12 void TPmatching(RV_Image* image){
13   CUfunction cuFunction;
14   CUfunction cuFunction2;
15   device.GetKernelHundle(&cuFunction, "TPmatching_kernel");
16   device.GetKernelHundle(&cuFunction2, "reduction_kernel");
17   cuParamSetTexRef(cuFunction,
18                    CU_PARAM_TR_DEFAULT,
19                    cuTexTPref);
20   result = image->cudaProcBox(&cuFunction,
21                               tp_image->Width,
22                               tp_image->Height,
23                               &cuFunction2);
24 }
25
26 int main(int argc, char* argv[]){
27   RV_Image* input_image = new RV_Image(argv[1]);
28   RV_Image* tp_image = new RV_Image(argv[2]);
29
30   device.RaCudaInit();
31   device.GetTexrefHundle(&cuTexTPref,"texTP");
32
33   tp_image->TexRefSetImage(&d_TPimage, &cuTexTPref);
34   TPmatching(input_image);
35   cutilDrvSafeCall(cuArrayDestroy(d_TPimage));
36   image->writerect(result.x,result.y);
37
38   device.RaCudaExit();
39   return 0;
40 }
```

Figure 13: Main program translated from Figure 12.

matching() is Figure 13 and TPmatching_kernel() in Figure 13.

First, the translator finds the invocation of procBox() at line 27 in Figure 12, and tries to translate the component function TPmatching(). The procBox() is one of the higher-order methods of RV_Image instance, and it is for appling a component function repeatedly inside a certain box defined by two coord arguments.

In the main program shown in Figure 13, TPmatching() is defined. It gets kernel hundlers for kernel functions, sets up texture reference, and passes kernel functions to the higher-order method cudaProcBox(), which is the CUDA-supported version of proxBox().

TPmatching() in Figure 13 is only a wrapper function, and the essence of TPmatching() is translated to TPmatching_kernel() in Figure 14. It calculates sum of absolute differences by calling the function SAD(). Now, SAD() is called from device code. Hence, __device__ qualifier is added to SAD().

Thread-local results are stored in the data4reduction[] array. In Figure 12, the variable *sad* is defined as a

```
/* kernel.cu (module) */

texture<int, 2, cudaReadModeElementType> texTP;
__device__ int SAD(int* idata, int wid, int hei,
                   int widBox, int heiBox, int x, int y){
  int sad = 0;
  int p1, p2;
  for(int j = 0; j < heiBox; j++){
    for(int i = 0; i < widBox; i++){
      p1 = idata[ (y + j) * w + (x + i) ];
      p2 = tex2D(texTP, i, j);
      int abs = absDiff(p1, p2);
      sad += abs;
    }
  }
  return sad;
}

extern "C"
__global__ void
TPmatching_kernel(int* idata, int4* data4reduction,
                  int wid, int hei, int widBox, int heiBox){
  int x = blockDim.x * blockIdx.x + threadIdx.x;
  int y = blockDim.y * blockIdx.y + threadIdx.y;
  int incX = gridDim.x * blockDim.x;
  int incY = gridDim.y * blockDim.y;
  int sad;
  int min = INT_MAX;
  for(int j = y; j < (hei - heiTP); j += incY){
    for(int i = x; i < (wid - widBox); i += incX){
      sad = SAD(idata, wid, hei, widBox, heiTP, i, j);
      if(sad < min){
        data4reduction[y * 256 + x].z = sad;
        data4reduction[y * 256 + x].x = i;
        data4reduction[y * 256 + x].y = j;
      }
    }
  }
}
```

Figure 14: Kernel module program translated from Figure 12.

global variable and overwritten in the component function TPmatching(). This lets the translator know that there needs a reduction operation for the variable *sad*. Hence, the code for reduction shown in Figure 15 is also generated.

The code in Figure 15 reduces the thread-local results. Gathering the data over threads on shared memory in each *Block*, the minimum value and its coordination is settled, and the process is repeated over all *Blocks* by *for* loop.

The code through the line 19 to 29 in Figure 15, sixteen threads in each *Block* access continuous addresses in shared memory. Hence, bank conflict and Warp divergence can be avoided.

```
/* reduction code */

extern "C"
__global__ void

reduction_kernel(int4* data4reduction, int4* g_odata){
  __shared__ int sdatax[256];
  __shared__ int sdatay[256];
  __shared__ int sdataz[256];

  // from Global Memory to Shared Memory
  unsigned int tid = threadIdx.x;
  unsigned int i = blockIdx.x * blockDim.x + threadIdx.x;
  sdatax[tid] = data4reduction[i].x;
  sdatay[tid] = data4reduction[i].y;
  sdataz[tid] = data4reduction[i].z;
  __syncthreads();

  // reduction operations on Shared Memory
  for(unsigned int s = blockDim.x / 2; s > 0;s >>= 1){
    if(tid < s){
      if(sdataz[tid] > sdataz[tid + s]){
        sdatax[tid] = sdatax[tid + s];
        sdatay[tid] = sdatay[tid + s];
        sdataz[tid] = sdataz[tid + s];
      }
    }
    __syncthreads();
  }

  if(tid == 0){
    g_odata[blockIdx.x].x = sdatax[0];
    g_odata[blockIdx.x].y = sdatay[0];
    g_odata[blockIdx.x].z = sdataz[0];
  }
}
```

Figure 15: Reduction operations generated from Figure 12.

Anisotropic 3D texture synthesis with application to volume rendering

Lasse Farnung Laursen
Technical University of Denmark
Richard Petersens Plads
Building 321
DK-2800 Kgs. Lyngby
lfla@imm.dtu.dk

Bjarne Kjær Ersbøll
Technical University of Denmark
Richard Petersens Plads
Building 305
DK-2800 Kgs. Lyngby
be@imm.dtu.dk

Jakob Andreas Bærentzen
Technical University of Denmark
Richard Petersens Plads
Building 321
DK-2800 Kgs. Lyngby
jab@imm.dtu.dk

ABSTRACT

We present a novel approach to improving volume rendering by using synthesized textures in combination with a custom transfer function.

First, we use existing knowledge to synthesize anisotropic solid textures to fit our volumetric data. As input to the synthesis method, we acquire high quality images using a 12.1 megapixel camera.

Next, we extend the volume rendering pipeline by creating a transfer function which yields not only color and opacity from the input intensity, but also texture coordinates for our synthesized 3D texture. Thus, we add texture to the volume rendered images. This method is applied to a high quality visualization of a pig carcass, where samples of meat, bone, and fat have been used to produce the anisotropic 3D textures.

Keywords

Volumetric Rendering, Texture Synthesis, Transfer function.

1. INTRODUCTION

The use of volumetric data is becoming increasingly common within research fields such as medical visualization, food production and graphics. This data is also ever increasing in size as the scanners providing the data, e.g. CT, MRI, and ultrasound scanners, are improving and thus able to provide higher resolutions. Increased precision and more detail is a natural evolution as having too much information, is somewhat of a luxury problem.

When concerned with rendering volumetric data in real time, two issues persist. Firstly, the features that we would like to visualize might be on a finer scale than the voxels, despite the ever increasing amount of volume data. In our case, we visualize pig meat, and the variation in the texture of pig meat is on a finer scale than the resolution of our CT scan. Moreover, the voxels in our CT scanned data are stretched ten times along one axis. This problem is compounded by a second issue which is the fact that the CT intensities represent material density, which is not directly related to the appearance of the underlying tissue.

We present a novel approach which aims to alleviate both issues. With prior knowledge about the type of volumetric data we wish to visualize, we synthesize an anisotropic 3D texture which is applied to the volume data via a customized transfer function. Using this transfer function, we map the CT intensities to a high resolution solid pig meat texture which gives a qualitatively far better representation of the meat than any single color. Moreover, the solid texture texels are not stretched.

Solid textures are an ideal fit when rendering volumetric data. In almost all cases, there is an interest in rendering what is beneath the surface or subdividing the data to expose some deeper layer. Since a solid texture shares the same number of dimensions as common volume data, its application is relatively straightforward.

2. Related Work

The focus of this paper can be divided into solid texture synthesis, and the application thereof in volumetric rendering.

Solid Texture Synthesis

Considerable work has been done within the field of texture synthesis, from parametric methods [HB95] to non-parametric methods [DB97, Har01], as well as alternative approaches [WL00]. Most texture synthesis algorithms use a sample texture as input, referred to from here on as exemplar. This exemplar forms the basis for either a parametric model, which synthesizes a new texture based on modeled

parameters, or for a non-parametric algorithm, which reuses elements from the exemplar and recombines these to create a new, yet similar, texture.

Solid texture synthesis has been pioneered and expanded upon within the past two decades. Several methods, both parametric [GD95] and non-parametric [Wei02], as well as alternate approaches [JDR04], have been presented.

A recent texture synthesis method, which we use to create our anisotropic textures, is called texture optimization [KEBK05, KFCO+07]. This method iteratively improves the texture as a whole, making each modification smaller and more refined.

Volumetric Transfer Function

Volume rendering [DCH88] has come a long way. Most applications today make use of graphics hardware to improve performance [CN94]. The field has seen a dramatic increase of research into all kinds of visualization techniques involving volumetric data. Most volumetric data originates from either computed tomography or magnetic resonance scans, which do not yield a direct mapping to appearance attributes (i.e. color and texture of the scanned tissue). An obvious field of research is therefore to provide proper color and texture to this otherwise appearance deficient data. The visible human project is one such example, where a male and female body has been scanned, and subsequently cut and photographed to obtain the correlation between density and appearance. One method with which to color the data, is the use of a transfer function [HKRs+06a].

Many methods for creating transfer functions exist. From a simple pre-defined function capable of transforming between two number domains, to a user defined transfer function allowing for iterative refinement through user input [CS07]. In most cases, user input is desirable since the transfer function is often used as a tool to highlight or hide specific features in the volume data.

Other approaches include Dong and Clapworthy [DC05], who use 2D input exemplars to apply and synthesize texture to a volumetric volume simultaneously. By analyzing the orientation of each voxel in the volume data a patch based synthesis strategy is applied to apply and expand the 2D exemplar to the volume.

Lu et al. [LEQ+07] expand upon an existing 2D synthesis algorithm to create a flexible system for volume illustration. By extending the concept of Wang Cubes into the third dimension Lu et al. create a tileable solid texture set.

Manke and Wünsche [MW09] provide a formal framework for applying solid textures to a volume, similar to the work in this paper. They also present methods for dealing with discontinuous mapping. In contrast to this paper, however they do not touch upon the scaling or periodicity issues of applying a repeating solid textures to a volume.

In this paper, we use a simple, piecewise constant transfer function which maps voxel intensities to entire texture volumes, similar to Manke and Wünsche [MW09]. Subsequently, the color values at the given position in the volume are obtained by lookup in these texture volumes. The voxel density is used as an indicator for opacity. The textures are applied in a multi-scale fashion to minimize the periodicity, which is further described in Section 6.

3. Overview

It has been our overall goal is to improve the visualization of CT scanned data. By applying a solid texture to the data via a transfer function, we are able to increase the visual detail at a minor cost to the computations required.

We employ the texture optimization method presented by Kopf et al. [KFCO+07], to synthesize our anisotropic textures. There is a large overlap with our description and [KFCO+07]. This is partly to highlight particular details of our implementation and partly to make the present paper more self-contained.

Unfortunately, the aforementioned texture synthesis method does a poor job of synthesizing textures with only low frequency features. This leads to some muscle textures being comparable to base noise textures with similar colors.

Due to computational limitations, synthesizing solids larger than 128x128x128 is not feasible. This presents a number of scale and periodicity issues which we explore in sections 6 and 7. In short, we apply the synthesized texture in multiple scales to allow for fine and rough effects. We still make use of the CT data to add additional rough detail.

The results of these iterative improvements are compared and discussed, also in section 7.

4. Solid Texture Synthesis

As previously explained, texture optimization is an iterative method where the difference between the input exemplar and the synthesized solid is minimized. The difference is measured by a global texture energy function which compares fixed sized 8x8 2D neighborhoods. For now, let us assume that each voxel/texel defines its own neighborhood. We define a simplified global texture energy function, similar to the one by Kopf et al. [KFCO+07].

$$E(N_s, N_e) = \sum_{i=1}^{c} \left\| N_{s,i} - N_{e,best} \right\|^r.$$

Equation 1: Global Energy Function.

The neighborhoods in the synthesized solid and input exemplar(s), are denoted by $\boldsymbol{N_s}$ and $\boldsymbol{N_e}$ respectively. The total number of number of neighborhoods from the synthesized solid ($\boldsymbol{N_s}$) is denoted $\boldsymbol{c}$. The ith vectorized neighborhood in the solid is denoted $\boldsymbol{N_{s,i}}$, and its closest match (in L_2 norm) in the input exemplar(s), is denoted by $\boldsymbol{N_{e,best}}$. The exponent $\boldsymbol{r} = 0.8$ makes the function more robust against outliers [KFCO+07, KEBK05].

Initially, the synthesized volume is comprised of randomly selected texels from the input exemplar. The volume is then iteratively improved to resemble the input exemplar(s). The process is comparable to an expectation maximization algorithm. We first find the "best looking" parameters, then we optimize based on those findings, and repeat the process.

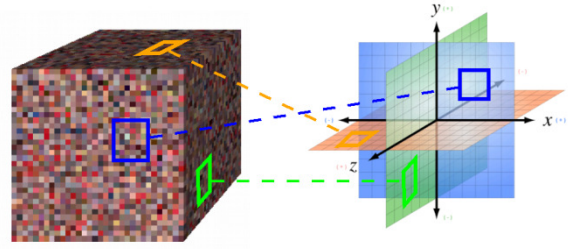


Figure 1: Exemplars on the three planes orthogonal to the main axes.

As mentioned previously, comparing the synthesized texture to the input exemplar(s) is done by comparing fixed sized 8x8 neighborhoods. These neighborhoods are extracted from both the synthesized volume and the input exemplar(s). However, there is not – as previously mentioned - one neighborhood assigned to each voxel. Rather, each voxel is indirectly related to the neighborhoods that includes it.

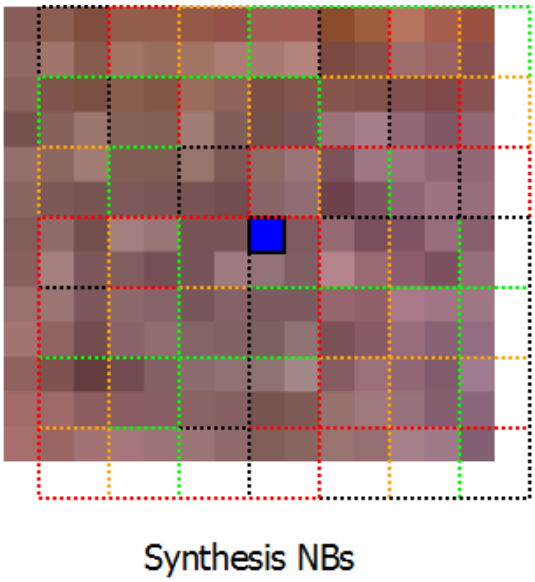


Figure 2: Density of neighborhoods on both exemplar and synthesis textures.

On the input exemplars, these neighborhoods lie on a densely populated grid, since we want to use all the available information provided to us, about the texture to be synthesized. In the synthesized volume the neighborhoods lie on a sparse grid (spaced 1 voxel apart like Kopf et al. [KFCO+07]), and only on planes orthogonal to the three main axes of our coordinate system, as shown in Figure 1. This serves to reduce computation time and avoid re-sampling issues.

Figure 2 visualizes the sparse grid upon which the synthesized neighborhoods lie. A given voxel – highlighted in blue - is a member of 16 on any given plane, due to the sythesized solids toroidal boundary conditions.

Once the all the neighborhoods have been extracted, the "best looking" parameters are then found by locating the least different neighborhood in the input exemplar(s), for each neighborhood in the synthesized volume. Once found, each voxel is assigned a new value based on texels in the corresponding best matches of the neighborhoods overlapping that voxel. Essentially averaging all the contributions to make a new color:

$$\boldsymbol{s_v} = \frac{\sum_{\boldsymbol{s_v} \in \boldsymbol{N_s}} \boldsymbol{t_{N_{e,best}}}}{\boldsymbol{c_{s_v}}}.$$

Equation 2: Voxel color calculation.

The new color assigned to the voxel in the synthesized solid, denoted $\boldsymbol{s_v}$, is an average of several existing color values. The above equation states that for each neighborhood $\boldsymbol{N_s}$ the voxel is a member of, we find the matching texel $\boldsymbol{t}$, in the best matching neighborhood $\boldsymbol{N_{e,best}}$. This sum is finally divided by $\boldsymbol{c_{s_v}}$, which denotes the number of neighborhoods the voxel $\boldsymbol{s_v}$ is a part of.

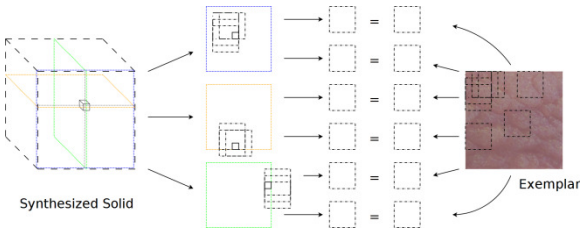


Figure 3: Neighborhoods on the three planes orthogonal to the main axes matched to input exemplar neighborhoods.

Figure 3 visualizes equation 2 in practice. In our sparsely populated grid on the synthesized solid, a single voxel is member of 16 neighborhoods on a single plane orthogonal to a main axis. Since we have three such planes, visualized as blue, yellow, and green in figure 3, the voxel is a member of a total of 48 neighborhoods. Each of these neighborhoods has a corresponding match in an exemplar. The texel overlapping the same position in each of these

neighborhoods contributes to the sum, which is eventually divided by the total number of contributions (in this case 48), yielding the new color.

The optimization algorithm is actually performed on multiple levels of differing quality. The synthesized solid initially consists of 32x32x32 voxels, and input exemplar(s) are scaled to 32x32 respectively. Once the synthesis process reaches certain conditions, outlined in section 4.5, the volume is scaled up to 64x64x64 using trilinear interpolation. Due to computational restrictions of performing a nearest neighbor search in a high dimensional space, the synthesis is only performed up to a resolution of 128x128x128.

Approximate Nearest Neighbor

In a standard-RGB texture, an 8x8 neighborhood consists of 192 values. Finding the nearest neighbor in a 192 dimensional space is a computationally expensive operation.

We apply the same optimizations as Kopf et al. [KFCO+07] to reduce the computation complexity. A principal component analysis is performed on the neighborhood vectors from the exemplar(s). By only preserving the coefficients required to maintain 95% of the variance, we can typically reduce the number of dimensions by half, or more.

We also employ the ANN: Approximate nearest neighbor library [MA10]. The library accepts a value ε, and returns an approximate nearest neighbor guaranteed to be at most $\varepsilon + 1$ away from the true nearest neighbor. We employ $\varepsilon = 2$ as dictated by Kopf et al. [KFCO+07].

Weighting Scheme

As previously mentioned in section 4, using an exponent of 0.8 in equation 1, causes it to be more robust against outliers. However, minimizing the L_1 norm is more cumbersome than minimizing the L_2 norm. So instead we introduce a weight into the equation and rewrite the terms of the energy function (1) to the following (similar to Kopf et al. [KFCO+07]):

$$\begin{aligned} &\left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{r} \\ &= \left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{r-2}\left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{2} \\ &= \boldsymbol{\omega_{e,best}}\left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{r} \end{aligned}$$

Equation 3: Energy function term re-write.

where $\boldsymbol{\omega_{e,best}} = \left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{r-2}$. This leads to the following quadratic formula which we seek to minimize:

$$\boldsymbol{E}\left(\boldsymbol{N_s}, \boldsymbol{N_e}\right) = \sum_{i=1}^{count(\boldsymbol{N_s})} \boldsymbol{\omega_{e,best}}\left\|\boldsymbol{N_{s,i}} - \boldsymbol{N_{e,best}}\right\|^{2}.$$

Equation 4: Improved energy function.

The weight parameter $\boldsymbol{\omega_{e,best}}$ makes sure that the exemplar neighborhood closest to a given synthesized neighborhood, carries the most weight. Instead of a straight average as applied in equation 2, we are now calculating a weighted average which leads to the following formula when calculating a new voxel value:

$$\boldsymbol{s_v} = \frac{\sum_{\boldsymbol{s_v} \in \boldsymbol{N_s}} \boldsymbol{\omega_{N_{e,best}}} \boldsymbol{t_{N_{e,best}}}}{\sum_{\boldsymbol{s_v} \in \boldsymbol{N_s}} \boldsymbol{\omega_{N_{e,best}}}}.$$

Equation 5: Weighted voxel color calculation.

Instead of dividing the sum by the total number of contributors, we now divide by the total amount of weight distributed among the contributions.

Meanshift

Although adjusting each contributing texel with a weight parameter yields better results and speeds up convergence, there are still numerous textures which fail to produces acceptable results. One persisting issue is that outliers still contribute to the final result, even if their contribution is minimal.

In order to minimize contribution from outliers, Kopf et al. [KFCO+07] employ a clustering approach, proposed by Wexler et al. [WSI07]. In short, every contributing texel is considered to be a cluster. These clusters are then merged depending whether their center is within a distance of τ to one another. If any new clusters emerge, the process of searching and merging is repeated, until no further clusters form. Only texels from the dominant cluster end up contributing to the new voxel value.

The threshold τ is decreased with each iteration over the course of a single resolution level convergence. Once the synthesized texture converges on a single level, the thresholding value τ is reset. We found that setting $\tau = 10$, $\tau = 0.05$, and $\tau = 0.01$ worked well in many cases, on the lowest, medium, and highest resolution level, respectively.

Histogram Matching

The previously mentioned modifications to the original synthesis method, speeds up convergence and minimizes the impact of outliers. However, the algorithm will occasionally converge at certain minima, which fail to make full use of the exemplar(s) details.

Kopf et al. [KFCO+07] address this issue by utilizing histogram matching. The weight each texel carries is further adjusted, based upon whether its contribution will increase, or decrease, the similarity between the histograms of the input exemplar, and the

synthesized solid. Practically, they achieve this by keeping track of a 16-bin histogram for each of the input exemplars' channels. Usually, this is just the red, green, and blue channel. Kopf et al. also note the importance of keeping this histogram up to date during each "maximization" phase. Otherwise, the method will just overshoot the intended histogram and overcompensate in the following iteration.

When synthesizing anisotropic textures we maintain one histogram per input exemplar. We let each contributing texel pull in the direction of its exemplars histogram, which seems to work well. Just like Kopf et al. we also traverse the voxels in a random order, to avoid any directional bias.

Histogram matching is an integral part of creating the best results possible via texture optimization. It makes the algorithm take global statistics into consideration while still allowing for the use of a small neighborhood window. Histogram matching also speeds up convergence significantly.

Synthesis Convergence Conditions

We found that a fixed number of iterations yielded the best result with most textures (J. Kopf, pers. comm.). Iterating 100 times on the lowest resolution, 20 on the next level, and 10 on the highest level, worked well with most textures.

5. Exemplar Acquisition

As with every other texture synthesis method, we require exemplars of the texture we intend to synthesize. Our exemplars were obtained using a 12.1 megapixel camera, Canon IXUS 120IS, in a well lit setting. Originally, we intended to obtain samples using a multispectral color and texture measurement vision system. This system measures up to 20 different bands across the visible and non-visible spectrum. These precise measurements are then combined to a final standard-RGB image. However, most household cameras actually yield more vivid and realistic colors as each sensor integrates a wider range of the spectrum than the more precise instrument.

6. Rendering

To visualize the volumetric data, a simple ray casting technique [HKRs+06b] is applied using the GPU. To obtain the start and end point for each ray, two rendering passes are performed of a cube showing the front- and backface respectively. The cube acts as our rendering proxy and yields the start and end position for each ray, which is recorded into a buffer using the fragment shader.

An additional rendering pass is then performed where the fragment shader traces a ray through the space enclosed by the cube. The ray is traced with 0.001 increments in relation to the unit cube around the volume, and accumulates more color and opacity as it traverses the volume. The "ray-color" starts off black and completely transparent. For each step through the volume, the current density is classified as air, skin, fat, meat, or bone, according to the Hounsfield scale [Sev04]. Its contribution to the overall "ray-color" as well as "remaining transparency" is calculated by the following formulas:

$$R_{rgb} = R_\alpha * l * D_{rgb} * D_\alpha$$
$$R_\alpha = R_\alpha * (1 - D_\alpha).$$

Equation 6: Color and transparency contribution per ray-step.

The contribution added to the existing color and transparency value of the ray is denoted as R_{rgb}. The amount of contributing light via simple lambertian shading [HKRs+06c], is denoted l. The contributing color and transparency from the classified density is denoted D_{rgb} and D_α respectively. When "ray-color" is completely opaque, the ray traversal is stopped.

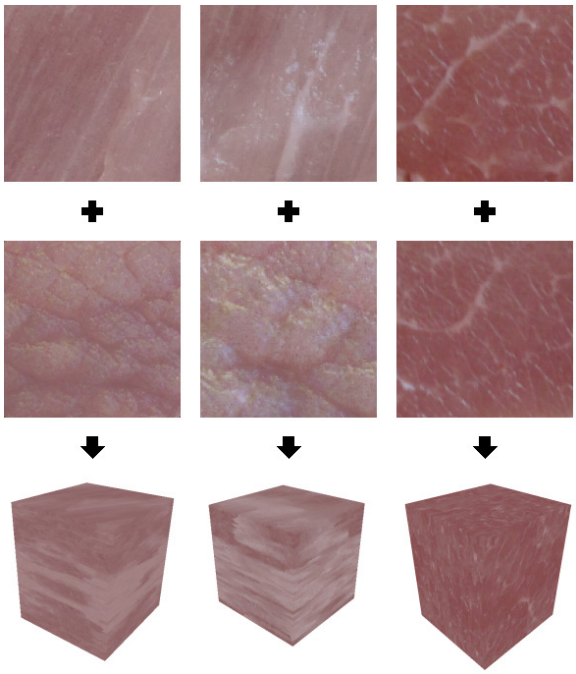


Figure 4: Three synthesized solids and their two input exemplars (pig muscle tissue). The left and middle synthesis' yield an unsatisfactory result.

Setting D_{rgb} in Equation 6 to the color from the appropriately scaled solid texture produces a result with significant periodicity at high magnification and almost uniform color at low magnification (because of mipmapping). This can be seen in Figure 5, in the top and bottom left. To ameliorate these issues, we combine the texture at three levels of scaling, and D_{rgb} is computed as illustrated in the next equation.

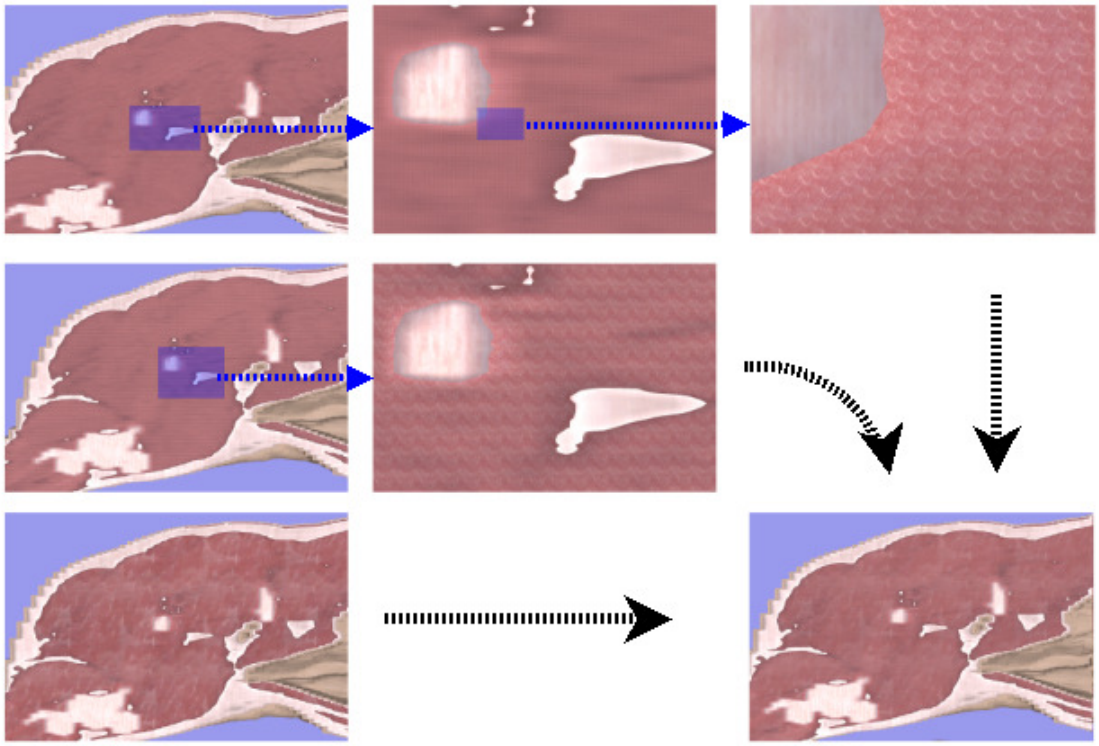


Figure 5: Three differently scaled muscle textures combined to create the final result. The top and middle segment show zoomed in areas to show finer detail.

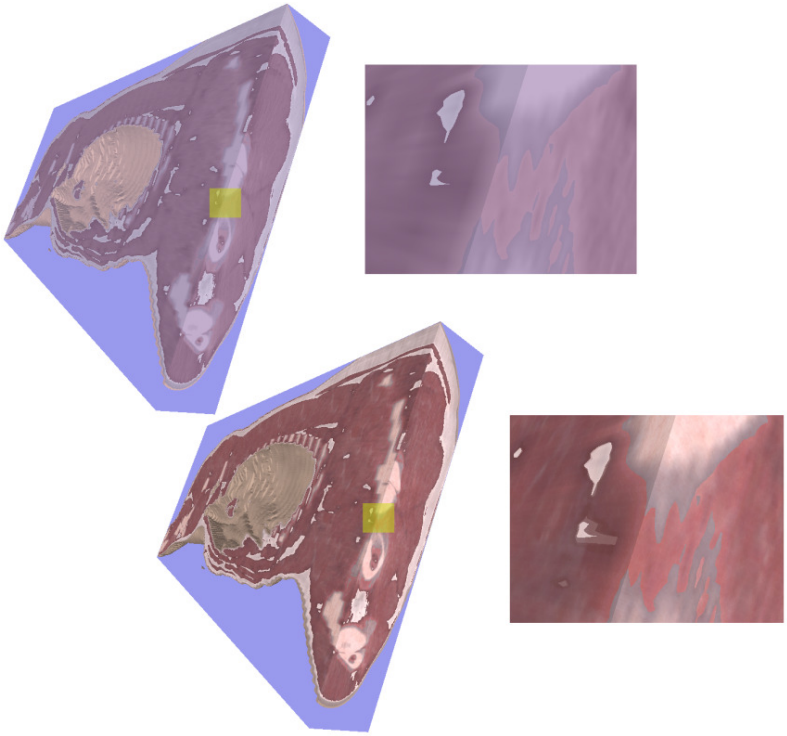


Figure 6: On the top, volumetric data from the pig carcass, visualized without enhanced graphics. The colors for the meat, bone, and fat tissue are the average color values of the textures applied on the right. On the bottom, volumetric data from the pig carcass, visualized with enhanced graphics. The highlighted sections in yellow indicate the zoomed section displayed on the right.

$$D_{rgb} = \frac{D^1_{rgb} f^1 + D^2_{rgb} f^2 + D^3_{rgb} f^3}{f^1 + f^2 + f^3}$$

$$i_{rgb} = D_{rgb} * int - t_{threshold}$$

$$D_{final} = D_{rgb} + i_{rgb}$$

Equation 7: Density case color contribution.

The color contribution consists of three differently scaled textures D^{1-3}_{rgb} and an associated weight factor f^{1-3}. For each tissue, the scales differ approximately a factor of 10, and the weight factor is always highest for the macro texture (approximately 3 to 1). Density is contributed to the final color value D_{final} via i_{rgb}. The value int represents a scaled measure of the density at that point in the volumetric data, and $t_{threshold}$denotes the density threshold of the contributing tissue. The result of combining the three synthesized muscle textures, along with the density modifier, is visualized in Figure 5.

An exception to the calculation outlined in Equation 7 is the skin color contribution which yields a constant average color of sampled pig skin, permeated by simplex noise [Per01] to give some variation to the surface.

The transparency value for either fat, muscle or bone is calculated via the following formula:

$$D_\alpha = \max(0.3, int + 0.25).$$

Skin has a constant translucency of 0.75, and air is completely transparent.

7. Results

We implemented the method described in this paper entirely in C++. The time required to generate a 128^3solid depends primarily on the size and richness of the input exemplars. On an Alienware m17x model (using only a single core) the synthesis of our three tissue types would usually converge after approximately 2-3 hours. It was our experience that the algorithm generated the best textures when only forcing two of the three dimensions to conform to input exemplars, regardless of whether isotropic, or anisotropic synthesis.

As previously mentioned in section 3, the synthesis algorithm has trouble synthesizing 3D textures based on input exemplars with primarily low frequency features. The two initial attempts in figure 4 show how the final synthesized solid ends up looking almost nothing like the original two textures used as input. The third synthesized solid is much more promising.

A question of scale arises when choosing how much surface a single exemplar should cover. To ensure we acquired as homogenous a sample as possible, and preserve detail, we chose to use exemplars covering a small area of approximately 2x2 cm.

As previously mentioned in section 6, the final color value of a voxel is modified by a simple mapping of the current density. The density modifier allows for a number of low frequency details to show, as is visualized in Figure 7.

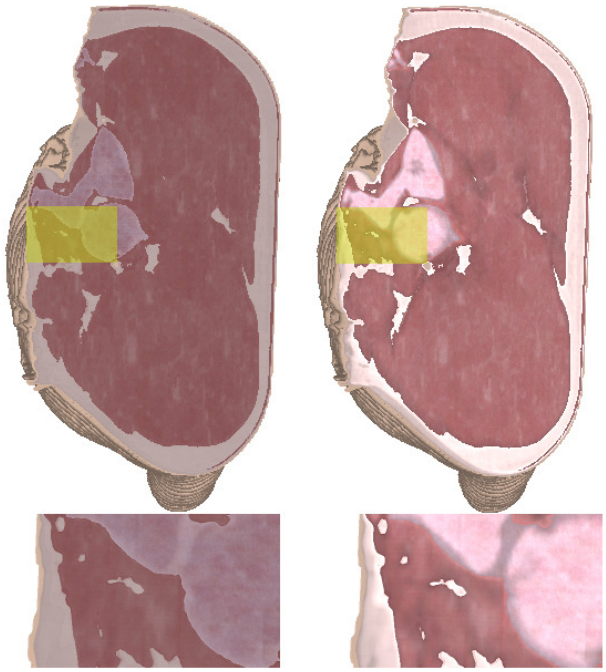


Figure 7: Two hams, with and without density value modification. The highlighted sections in yellow indicate the zoomed section displayed on the bottom.

The final result of the visualized volume data with all the aforementioned techniques applied is shown in Figure 6.

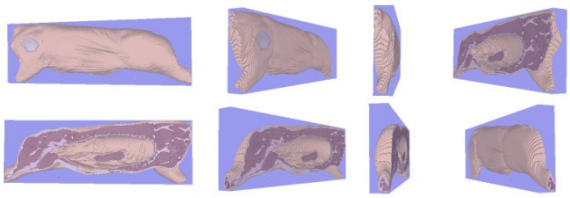


Figure 8: The volume data rotation pattern of the preliminary benchmark. Unenhanced pig visualized.

We perform a preliminary benchmark of the applied synthetic textures by rotating the volume one complete turn, around the y-axis, as seen in Figure 8.

	Std. Graphics	Enh. Graphics
Min. Fps	53	47
Max. Fps	118	102
Avrg. Fps	76.817	61.117

Table 1: Preliminary performance measurements.

The application of the synthesized textures only requires three additional texture lookups per

visualized voxel. Since texture lookups are implemented on the hardware level, it comes as no surprise that the performance loss is minimal, as seen in Table 1. However, creating the synthesized textures is another matter. Due to the complexity of a nearest neighbor search in a high dimensional space, performing the synthesis in real-time is an impossibility.

8. Conclusions and Future Work

We have utilized an existing texture synthesis approach to produce three anisotropic textures, which were then applied to volumetric data via a custom transfer function, improving upon the original colorless data.

The technique can potentially be applied to any type of volumetric data and is not necessarily constricted to organic tissue.

Although the result has improved significantly, there is still room for improvement.

Figure 9: A close up of the visualized volumetric data showing computed tomography artifacts.

Our light model is simplistic. Better modeling of how light and meat interact would be an obvious next step since, recently, techniques for real-time interactive computation of translucent surfaces have started to appear, e.g. [WWH+10].

As mentioned in the previous section, we modify the final color slightly via the density of the volumetric data. While this adds significant detail to the final visualization, it also introduces a number of artifacts introduced by the scanning method. Figure 9 shows how the data acquisition rays from computed tomography leaves visible artifacts in the volume data.

Due to time required to perform a complete solid texture synthesis it could be advantageous to create a larger pre-computed library of multiple tissue types (in addition to the three described in this paper).

Theoretically, it would also be possible to synthesize in-between textures by using an input exemplar from each tissue type, to smooth the transition between them. A few practical experiments are required to see how convincing the resulting solid textures would be.

It would also be interesting to implement and compare the technique demonstrated by Lu et al. [LEQ+07]. Using their extension to the wang cube model is also a way of avoiding periodicity in the applied texture.

9. ACKNOWLEDGMENTS

We would like to thank Johannes Kopf for the invaluable correspondence during the development of this paper. We also extend our thanks to the anonymous reviewers in helping us improve on the paper. This research was supported in part by the Danish Meat Research Institute. The CT scan of the pig carcass was also kindly provided by the Danish Meat Research Institute.

10. REFERENCES

[CJ06] Thierry Carrard and Manuel Juliachs, Bandwidth-efficient Hardware-Based Volume Rendering for Large Unstructured Meshes, WSCG 2006, 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, CZECH REPUBLIC, JAN 30-FEB 03, 2006, pp. 169–176 (English).

[CN94] Timothy J. Cullip and Ulrich Neumann, Accelerating volume reconstruction with 3d texture hardware, Tech. report, Chapel Hill, NC, USA, 1994.

[CS07] Amit Chourasia and Juergen R. Schulze, Data Centric Transfer Functions for High Dynamic Range Volume Data, WSCG 2007, 15th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen, CZECH REPUBLIC, JAN 29-FEB 01, 2007, pp. 9–15 (English).

[DB97] Jeremy S. De Bonet, Multiresolution sampling procedure for analysis and synthesis of texture images, SIGGRAPH ’97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques (New York, NY,

USA), ACM Press/Addison-Wesley Publishing Co., 1997, pp. 361–368.

[DC05] Feng Dong and Gordon J. Clapworthy, Volumetric texture synthesis for nonphotorealistic volume rendering of medical data, The Visual Computer 21 (2005), 463–473.

[DCH88] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan, Volume rendering, SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM, 1988, pp. 65–74.

[GD95] D. Ghazanfarpour and J. M. Dischler, Spectral analysis for automatic 3-d texture generation, Computers & Graphics 19 (1995), no. 3, 413 – 422.

[Har01] P Harrison, A non-hierarchical procedure for re-synthesis of complex textures, WSCG ' 2001, 9th International Conference on Computer Graphics, Visualization and Computer Vision, PLZEN, CZECH REPUBLIC, FEB 05-09, 2001, pp. 190–197 (English).

[HB95] David J. Heeger and James R. Bergen, Pyramid-based texture analysis/synthesis, SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM, 1995, pp. 229–238.

[HKRs+06a] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel, Real-time volume graphics, pp. 81–102, A. K. Peters, Ltd., Natick, MA, USA, 2006.

[HKRs+06b] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel, Real-time volume graphics,pp. 163–185, A. K. Peters, Ltd., Natick, MA, USA, 2006.

[HKRs+06c] Markus Hadwiger, Joe M. Kniss, Christof Rezk-salama, Daniel Weiskopf, and Klaus Engel, Real-time volume graphics,pp. 114–116, A. K. Peters, Ltd., Natick, MA, USA, 2006.

[JDR04] Robert Jagnow, Julie Dorsey, and Holly Rushmeier, Stereological techniques for solid textures, ACM Trans. Graph. 23 (2004), no. 3, 329–335.

[KEBK05] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra, Texture optimization for example-based synthesis, SIGGRAPH '05: ACM SIGGRAPH 2005 Papers (New York, NY, USA), ACM, 2005, pp. 795–802.

[KFCO+07] Johannes Kopf, Chi-Wing Fu, Daniel Cohen-Or, Oliver Deussen, Dani Lischinski, and Tien-Tsin Wong, Solid texture synthesis from 2d exemplars, ACM Transactions on Graphics (Proceedings of SIGGRAPH 2007) 26 (2007), no. 3, 2:1–2:9.

[LEQ+07] Aidong Lu, David S. Ebert, Wei Qiao, Martin Kraus, and Benjamin Mora, Volume illustration using wang cubes, ACM Trans. Graph. 26 (2007).

[MA10] David M. Mount and Sunil Arya, Ann: A library for approximate nearest neighbor searching., http://www.cs.umd.edu/ mount/ANN/, 2010.

[MW09] Felix Manke and Burkhard C.Wuensche, Texture-enhanced direct volume rendering, Proceedings of the 4th International Conference on Computer Graphics Theory and Applications (GRAPP 2009) (Lisbon, Portugal), 2009, pp. 185–190.

[Per01] Ken Perlin, Noise hardware, In Real-Time Shading SIGGRAPH Course Notes (2001), Olano M., (Ed.)., 2001.

[RV06] Daniel Ruijters and Anna Vilanova, Optimizing GPU Volume Rendering, JOURNAL OF WSCG, 14th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision, Plzen Bory, CZECH REPUBLIC, JAN 30-FEB 03, 2006, pp. 9–16 (English).

[SC07] Sakurambo and John Cayley, 3d coordinate system, Image on the Internet, 2007.

[Sev04] El Sevier, Introduction to ct physics, http://www.angelfire.com/nd/hussainpassu/Physics_of_Computed_Tomography.pdf, 2004.

[Wei02] Li-Yi Wei, Texture synthesis by fixed neighborhood searching, Ph.D. thesis, Stanford, CA, USA, 2002, Adviser- Levoy, Marc.

[WL00] Li-Yi Wei and Marc Levoy, Fast texture synthesis using tree-structured vector quantization, SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques (New York, NY, USA), ACM Press/Addison-Wesley Publishing Co., 2000, pp. 479–488.

[WSI07] Yonatan Wexler, Eli Shechtman, and Michal Irani, Space-time completion of video, IEEE Trans. Pattern Anal. Mach. Intell. 29 (2007), no. 3, 463–476.

[WWH+10] Yajun Wang, Jiaping Wang, Nicolas Holzschuch, Kartic Subr, Jun-Hai Yong, and Baining Guo, Real-time rendering of heterogeneous translucent objects with arbitrary shapes, Computer Graphics Forum 29 (2010), 497–506.

Using an Augmented Reality game to find matching pairs

M. Carmen Juan; Marta Carrizo; Francisco Abad
Instituto ai2
Universitat Politècnica de València
Camino de Vera, s/n. 46022
Valencia, Spain

Miguelón Giménez
Escola d'Estiu
Universitat Politècnica de València
Camino de Vera, s/n. 46022
Valencia, Spain

ABSTRACT

In this paper we present an Augmented Reality (AR) game for finding matching pairs to learn about endangered animals in a fun way. Thirty-one children participated in a study. These children played the AR game and the equivalent real game. We have compared the results of the two games. We have evaluated different aspects (technical, orientational, affective, cognitive and pedagogical). The results indicate that children enjoyed playing the AR game more than playing the real game and that they perceived the AR game to be more fun than the real game. The children preferred the AR game to the real one and also seemed to learn about the subject of endangered animals.

Keywords

Augmented Reality, edutainment, finding pairs.

1. INTRODUCTION

In this paper, we present an Augmented Reality (AR) game for finding matching pairs. In an AR system, users see an image composed of a real image and virtual elements that are superimposed over it. The most important aspect in AR is that the virtual elements add relevant and helpful information to the real scene.

Our AR game follows the rules and appearance of the popular pair game. Since the game uses AR, over the pieces of the game can appear images as well as explanatory videos about the endangered animals. The animals and part of the information related to them were chosen from the Red List of Threatened species (http://www. iucnredlist.org) published by the International Union for the Conservation of Nature and Natural Resources (IUCN). This list was created in 1963 and is the world's most comprehensive inventory of the global conservation status of plant and animal species. The information on the Red List is updated on the web site whenever possible (annually). A full analysis of the data on the Red List is published once every four years. There are nine categories on the IUCN Red List: Extinct, Extinct in the Wild, Critically Endangered, Endangered, Vulnerable, Near Threatened, Least Concern, Data Deficient, and Not Evaluated. In this paper only two of these categories are described (critically endangered and vulnerable). Critically endangered, is defined as a species that is facing an 'extremely high risk' of extinction in the wild. Vulnerable, is defined as a species that is facing a 'high risk' of extinction in the wild.

The main objective of this work was to develop an innovative AR system to allow children to learn about the animals that are at risk of extinction in a fun way. The system is fun because it is played as a game. It is innovative because as far as we know there is no other AR system that has been developed for this purpose. Another objective was to evaluate different aspects of the AR game.

Taking into account the multidimensionality of learning as well as AR as a field, there are a number of technical, orientational, affective, cognitive, pedagogical and other aspects that can be considered in the evaluation. The technical aspect examines usability issues, regarding interface, physical problems, and system hardware and software. The orientation aspect focuses on the relationship of the user and the augmented environment; it includes navigation, spatial orientation, presence and immersion, and feedback issues. The affective

parameter evaluates the user's engagement, likes and dislikes, and confidence in the virtual environment. The cognitive aspect identifies any improvement of the subject's internal concepts through this learning experience. Finally, the pedagogical aspect concerns the teaching approach: how to effectively gain knowledge about the environment and the concepts that are being taught.

This paper is organized as follow. Section 2 focuses on AR systems that have been used for learning. Section 3 presents our AR system and includes the software and hardware requirements as well as a description of the game. Section 4 presents the results of the game evaluation for the different aspects: technical, orientational, affective, cognitive and pedagogical. Finally, in section 6, we present our conclusions, our suggestions for improvements and future work.

2. RELATED WORK

Our work is not the first application for learning. Learning is one of the fields were AR has already been applied. For example, HIT Lab NZ (www.hitlabnz.org), University of Canterbury, New Zealand has developed several AR systems. The first one was The Magic Book [Bil01]. The Magic Book was presented as an example of an ARToolKit application. It looked like a normal book, but there were markers on the pages. A marker is a white square with a black border inside that contains symbols or letter/s. When the system recognized a marker, an image was shown or a story was started. Books of this type can be used for other purposes. A second work presented by this group was the S.O.L.A.R system. It was created for the TeManawa Science Centre (Palmerston North, New Zealand). It was an AR system for learning the position of each planet in the Solar System [Woo04]. A third work that is worthy of mention is the AR Volcano. It was developed for Science Alive! (Christchurch, New Zealand). It was a system for learning about volcanoes [Woo04]. Another work developed by this group was the BlackMagic. It was developed for the Telecom Technology Pavilion at the America's cup in New Zealand in 2003. It was a MagicBook that told the history of the America's Cup [Woo04].

Another research group that has also developed several AR systems in this field is the Mixed Reality Lab of Singapore (www.mixedrealitylab.org). They have developed several AR systems which include: the sun system, how plants grow and the Magic Story Cube. In the sun system, several concepts that are related to the solar system were explained. In the plant system children learned how plants germinate, disperse, reproduce and perform photosynthesis. The Magic Story Cube used a cube as a tangible interface that was folded or unfolded and, depending on the markers that were visible, the story was different. The Magic Story Cube presented the story of Noah's ark.

Other groups have also been working on the development of different AR systems. For example, Bimber et al. [Bim01] presented the Virtual Showcase. It placed virtual objects on real artefacts. One of the most outstanding applications was to place skin and bones on the skull of a Raptor dinosaur. Shelton & Hedley [She02] developed an AR system to teach the relation between the earth and the sun to geography students. In 2004, Kaufmann [Kau04] presented Construct3D as his PhD dissertation thesis. Construct3D was an AR system for constructing 3D geometries. It was designed to teach mathematics and geometry. Construct3D was tested with 14 students from two high schools in Vienna. The results from two evaluations showed that Construct3D was easy to use, required little time to learn, and encouraged learners to explore geometry. Larsen et al. [Lar05] presented an AR system for learning how to play billiards. The most outstanding characteristic of this system was that the game was played on a real billiard table. Organic chemistry can also be taught using an AR system [Fje07]. Fjeld's system, users interacted directly with 3D molecular models. In 2008, Sykora et al. [Syk08] presented a colour ball tracking that was used for direct manipulation with real objects. They presented two learning applications. The first one for learning basic principles of chemical reactions. Color balls were used to represent atoms. They combined typical AR markers with the color ball tracking that had a special semantic meaning. The second one for learning organs in a human body where the balls were used as a pointing device.

Our work is neither the first work that compared different presentation forms, for example Despina et al. [Des10] compared six different types of museum exhibits, one traditional and five interactive ICT exhibits. The exhibits were: a traditional map learning activity, a virtual tour projection, a multi-touch table application and three different AR applications (AR puzzle, AR map and Touch History). They evaluated the experience of young users with the exhibits. They included two questions. From the question: "your experience from the exhibit was (awful, not very good, good, really good, and brilliant)". Related to the brilliant score category, the touch table scored 76%, followed by the AR puzzle with a score of 67%, followed by the Virtual Reality tour and AR Map (with scores near 50%). They concluded that the experience scores top marks for the interactive ICT systems.

3. DESCRIPTION OF THE AR GAME

In our AR game, over the markers appeared images (Figure 1) and videos of endangered animals such as the Iberian lynx. The videos of the animals described the physical characteristics of the animal, its habitat and food, and also explained the causes of its possible extinction. The animals and the categories that were included in the game were the following:

- Critically endangered: Iberian lynx, Lowland gorilla, Red wolf, Orinoco crocodile, and Javan rhinoceros.
- Vulnerable: Polar bear, Iberian eagle, Humpback whale, and Amazonian manatee.

Figure 1. Looking for a pair

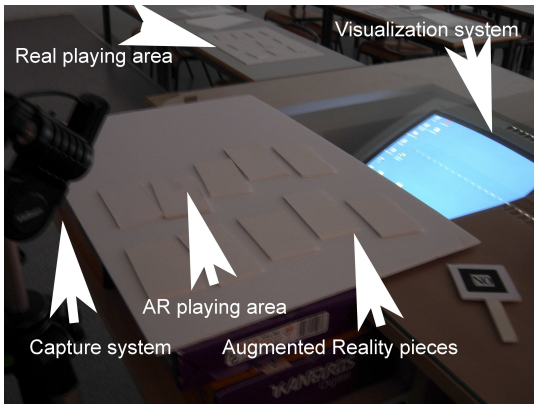


Figure 2. Elements used in the game

For the interaction with the game, the child only has to use markers with different symbols in their interior. In our work, we have not used a direct augmentation. The markers are a kind of a "remote control", but they are not directly augmented. The augmentation can be seen on the screen next to the playing area. That is, the child can see the real markers in front of him/her (playing area) and next to it, the screen with the augmented scene. Figure 2 shows the elements used in the game. The basic steps in the AR game are:

1) Initialization of the video entry and download of the files that contain the pattern and camera data, the XML files containing information related to the animals that are going to be shown.
2) The game asks the child to find the first animal pair. The child turns over one piece and then turns another one over.
3) The system identifies the visible markers and shows the related animal over them. The person in charge of the test must make sure that the child only turns over two pieces at a time. If the two markers belong to the desired animal, the game detects this situation and congratulates the child by telling that s/he has found the right animal. If the markers do not match, the child must continue to turn pieces over. Figure 1 shows an image where the child did not find a pair and s/he had to continue turning pieces over. The children could hold the pieces in their hands and look closely at the images.
4) If the child finds the right animal, the game asks if s/he wants to know more about the animal. The child has to use a marker with 'Yes' in its interior for answering yes, and a marker with 'No' in its interior for answering no. S/he has to place the chosen marker in a visible area in order to continue with the game. To facilitate the interaction, the child used a palette with 'No' in one side and 'Yes' on the another side. This palette can be seen in the left-lower area of Figure 2. If the answer is yes, the game shows a video over the visible marker/s. It shows the characteristics of the animal and explains the causes for its possible extinction.
5) The child can skip the rest of the video by using a marker with the symbol "*" at any point. For using this symbol, the child used another palette.
6) The game asks if the child wants to search for another animal. If the answer is yes, the game repeats step 2; if the answer is no, the game ends. The way of answering is the same as in step 4 (marker with 'yes'/'no').
7) At the end of the game, the child receives a score that depends on the number of animals successfully matched and the amount of time. The greater the number of matched pairs and the lower the time, the higher the score. The children's score is then compared with the ten best scores that are stored in an XML file.

In order to be able to extend the game to other themes with minimum changes, we included as much information as possible in XML external files. We used two different kinds of XML files. One of them contained the identification number of the image, the name of the animal, the length of the video, and the path to obtain the related images and videos. Another

XML file contained the total number of pairs available and also which ones were going to be used in each game. Another XML file was used to store the children's scores. For our game, we had a total of 10 animals. The ten markers used for showing animals are depicted in Figure 3. Figure 4 shows a boy playing with the AR game.

In order to validate the AR game, we compare it with a real game. The basic steps for playing with the real game are the following:

1) The child sits in front of the table for playing the real game (Figure 2, real playing area). The person in charge of the validation asks the participant to find a pair. The participant uses real pieces to find a pair. Figure 5 shows the pieces of the real game.
2) The participant turns over two pieces to find a pair. If the participant does not find a pair, the person in charge of the validation tells the participant to try again. If the participant finds a pair, the person in charge of the validation asks if the participant wants to know more about the animal. If the answer is no (verbally), the game goes to step, 3. If the answer is yes (verbally), the person in charge of the validation shows a page with images and text. The text is the same as the narrative of the video that is reproduced in the AR game. It explains the characteristics of the animal, its habitat and food, and it also explains the causes of the animal's possible extinction.
3) The person in charge of the validation asks if the participant wants to search for another pair. If the answer is yes, the game repeats step 1; if the answer is no, the game ends.

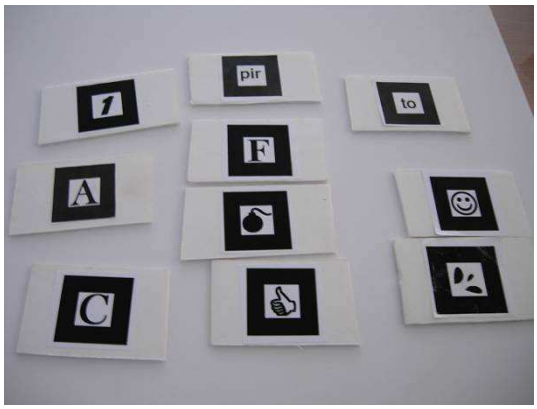


Figure 3. Markers used in the game

Figure 4. A boy is watching the video of the Iberian lynx

To capture the video, we used QuickCam Pro for Notebooks. The camera was fixed to a tripod which was placed next to the child. We used a table with back-projection as visualization system (a table made of glass under which a CRT monitor was placed). Figure 2 shows this table and its location.

To develop the system, we used the OsgART library (www.artoolworks.com/community/osgart). It is a C++ library that allows developers to build AR applications using the rendering capabilities of Open Scene Graph (OSG) and the tracking and registration algorithms of ARToolKit [Kat99]. OSG is a set of open source libraries that primarily provides scene management and graphics rendering optimization functionality to applications. It is written in portable ANSI C++ and uses the standard OpenGL low-level graphics API. ARToolKit is an open source vision tracking library that allows a wide range of AR applications to be easily developed. The required elements for the application are: a USB or FireWire camera, and a marker.

The animals' videos used in the game used AVI format. Their length ranged from 45 second to 1 minute. The animals' images were saved using the JPEG image file format.

Figure 5. The pieces of the real game

4. STUDY AND RESULTS

As stated in the section 1, one of the objectives of this work was to evaluate different aspects of the AR game: technical, orientational, affective, cognitive and pedagogical. To do this, we compared subjective measures taken in a real game and in the AR game.

The study included 31 children, 17 boys and 14 girls (aged from 6 to 12 years old, mean=7.7, SD=2.1). The children's parents signed an agreement to allow them to participate in this study

Children were counterbalanced and assigned to one of two conditions: a) Children who used the real game first and then the AR game, 15 children; b) Children who used the AR game first and then the real game, 16 children.

The protocol was the following. Before using either game the children were asked to fill out an entry questionnaire (appendix, Table 3). Then, the children were shown an explanatory video about the Red List of threatened species of the IUCN and also told how to play the games. This part was easy because most of the children already knew how to play to this popular matching game. The children then played the first game. After the game, the children were asked to fill out a post-game questionnaire (appendix, Table 4) and a reduced version of the presence questionnaire (appendix, Table 5) by Slater et al. [Sla94]. After filling out the two questionnaires the children played the second game. After playing, the children were again asked to fill out the post-game questionnaire and the same presence questionnaire. Finally, they were asked to fill out a final questionnaire (appendix, Table 6). The children played with the AR game at about 15 minutes and with the real game at about 10 minutes. All the questionnaires had to be answered on a scale from 1 (not at all) to 7 (very much).

The significance level was set to 0.05 in all tests. Table 1 shows paired t-tests for the scores given to the post-game questionnaire after playing both games. As this table shows, there was a statistical difference for questions 1 to 4. This indicates that children enjoyed playing the AR game more than playing the real game. They perceived the AR game as being more fun than the real game. Question 4 for the perceived value indicates that children preferred the AR game. Question 5 was also included to determine the perceived value, and there was no statistical difference between the two games. On the other hand, the children perceived the real game as being easier to play. There was no statistical difference between the two games for questions 5 to 9, indicating that the two games induced similar motivation and intention to change.

	AG1	AG2	AG3	AG4	AG5
AR	6.74(0.77)	6.58(1.23)	5.74(1.63)	6.55(1.23)	6.29(1.53)
Real	6.06(1.00)	5.90(1.25)	6.77(0.50)	5.90(1.51)	6.19(1.42)
t	4.33**	3.02**	-3.79**	3.07**	0.45
p	<0.001**	0.005**	0.001**	0.005**	0.655

	AG6	AG7	AG8	AG9
AR	6.97(0.18)	6.87(0.34)	5.90(1.70)	6.42(0.81)
Real	6.94(0.25)	6.84(0.37)	5.84(1.66)	6.36(0.80)
t	1	0.57	0.57	1.44
p	0.325	0.572	0.572	0.161

Table 1. Means (SD) of the AR game and the real game, and paired t-test of the post-game questionnaire, d.f. 30, **' indicates significant differences

In order to determine whether or not the order of play had an effect on the scores in the second game, the sample was divided into two groups (children who used the real game first and children who used the AR game first) and Student t tests for the scores given to all questions were applied. No significant statistical differences were found, this indicates that the order of play did not influence the children's scores for the post-game questionnaire.

To determine the level of perceived learning we compared the initial score for the children's knowledge about the animals that are at risk of extinction and the causes (I1, mean(SD)=3.45(1.183)) with the perceived learning scores after playing the two games (A2P1, mean(SD)=6.10(0.98)). Using paired t-test, t(30)=-12.90, p<0.001, the results show that there was a significant statistical difference between the two scores. The data indicate that children seem to learn using the games.

We analyzed the questions that related to the children's attitude using paired t-tests. We used I2 and AG6. Our analysis starts with the first group that used the AR game first and then the real one. In this case, the initial score for question I2 was very high, mean(SD)=6.69(0.79). This implies that even before playing either game the children thought we should provide greater protection to animals that are at risk of extinction in order to prevent their extinction. We compared the initial values with the values given after playing a game (AR/real). For the AR game, mean(SD)=7.00(0.00), t(15)=-1.58, p=0.136. The

data show there is no statistical difference between the two values. The mean and standard deviation after playing the real game (after the AR game) was exactly the same. With regard to the second group, that is, children who played the real game first and then the AR one, the initial score was also very high for question I2, mean(SD)=6.87(0.35). Playing the AR game second, the values are the following, mean(SD)=6.93(0.26), t(14)=-1, p=0.334. The mean and standard deviation after using the real game first was the same as the initial score. Again, there was no statistical difference for the group who played the AR game after the real game.

We analyzed the questions related to the children's motivation to change. We used I3 and AG7, and paired t-tests. Our analysis starts with the first group that played the AR game first and then the real one. In this case, the initial score for question I3 was very high, mean(SD)= 6.56(0.89). As in previous analysis, even before playing either game, the children were willing to support initiatives to protect animals that are at risk of extinction (AG7). We compared the initial values with the values given after playing a game (AR/real). Playing the AR game first, the values are the following, t(15)=-1.70, p=0.111. As can be deduced from the data, there is no statistical difference between the two values. Again, as in previous analysis, the mean and standard deviation after playing the real game second was exactly the same. For the children who played the real game first and then the AR one, the initial score was also very high for question I3, mean(SD)=6.87(0.35). After playing the real game first, the values were: mean(SD)=6.73(0.46), t(14)= 1.47, p=0.164. After playing the AR game second, the values were: mean(SD)=6.80(0.41), t(14)=1.00, p=0.334. For the second group there was no statistical difference since the initial value was so high, the values after playing both games were slightly lower.

In our study, we used two questions for the sense of presence (the presence score is taken as the number of answers that have a score of 6 or 7). The scoring was on a scale of 1-7. The SUS Count indicates the mean of the test count of scores of 6 or 7 for the 2 questions. The SUS Mean uses the mean score across the 2 questions instead. For the AR game, these values are: SUS Count=1.90(0.40), SUS Mean=6.69(0.64). From these data, it is possible to deduce that the AR game induces a great sense of presence. Table 2 presents the rest of the data for the presence questionnaire. It shows paired t-tests for the scores given after playing the two games. The analysis of the data indicates there is no significant statistical difference between the two games. This implies that children perceived the AR game as being real. In order to determine whether or not the order of play had effect on the scores in the second game, the sample was divided into two groups (the group of children who played the real game first and the group of children who played the AR game first). Student t tests for the scores given to all questions were applied. No significant statistical differences were found. Therefore, the order of play did not influence the children's scores for the presence questionnaire.

Figure 6 presents the results for the question AP2. Figure 7 shows children's preferences grouped by age. The majority of the children preferred the AR game. For the older children, this percentage was higher. Several explanations that the children gave for preferring the AR game were: 1) There were videos; 2) I could move the videos on the computer; 3) The videos explained much better why the animals are at risk of extinction; 4) You learn more with the videos; 5) Because I really liked the videos that I saw; 6) Because I could see my hands in the image. However, there were some children who liked the real game better. For the following: Because it was easier.

	P1	P2
AR	6.71(0.64)	6.68(0.70)
Real	6.90(0.30)	6.87(0.34)
t	-1.99	-1.65
P	0.056	0.110

Table 2. Means (SD) of the AR game and the real game, and paired t-tests for scores given to the presence questionnaire after playing the two games, d.f. 30

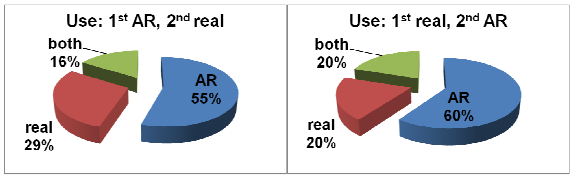


Figure 6. Children's preferences

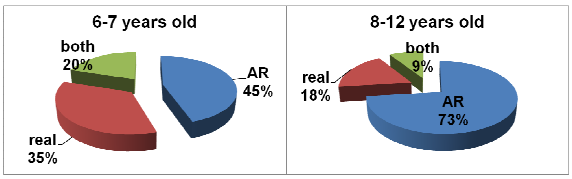


Figure 7. Children's preferences grouped by age

Some positive comments related to the AR game were the following: 1) I will talk with my sister who is a biologist and I will tell her about everything that I have learned; 2) This game has surprised me; 3) I enjoyed seeing the images and videos presented this way.

The only negative comments were the following: 1) I am not interested in the videos and I do not want to listen; 2) I do not like the game because for me it is

not a game, it is another way of learning; 3) I don't feel like playing.

5. CONCLUSIONS

We have presented an AR game that implements a popular pair game for learning about different animals that are at risk of extinction. The children learn the animals' habits, characteristics, and the causes of possible extinction. Thirty-one children played the AR game and the equivalent real game. To our knowledge, this is the first AR game with these characteristics that has been developed and evaluated for learning.

We have evaluated the aspects that are normally used in the evaluation of educational systems (technical, orientational, affective, cognitive and pedagogical). The results indicate that children enjoyed playing the AR game more than playing the real game and that they perceived the AR game to be more fun than the real game. With regard to presence, the questionnaires indicated that the AR game induced sense of presence in children and that this sense of presence was similar to what they felt in a real environment. Analyzing preference by age, it can be deduced that older children liked the AR game more than the younger ones. If attitude and motivation to change are considerer, the results indicate the following. Before playing either game the children thought 'We should provide greater protection to animals that are at risk of extinction in order to prevent their extinction'. In this case, the results indicate that the children's attitude did not change after playing the games. Before playing either game, the children were also willing to support initiatives to protect animals that are at risk of extinction. As in the previous case, the results indicate that the children's motivation to change did not change.

These results are encouraging, because AR has demonstrated that the children have fun and enjoyment; and induce sense of presence. Also before using either game, the children thought that "We should provide greater protection to animals that are at risk of extinction in order to prevent their extinction". In spite of this, they perceived more value in the AR game than in the real game.

More work has to be done to evaluate educational AR systems. We have evaluated some parameters of the different evaluation aspects, but a more exhaustive evaluation could be performed. This would provide a more significant contribution to educational systems, particularly AR educational systems.

The system can be improved in several ways. First of all, the glass surface on the table reflected. This problem could be solved using a non-reflecting glass surface or by eliminating it completely. We placed the camera on a tripod next to the child, but a more stable element could be used instead. Second, another improvement that would involve greater changes is to convert the system from 2D to a 3D version. Related to the 3D version, if models of 3D animals with a significant number of polygons were used, the rendering speed would be an important aspect to evaluate. In that case, modern Graphics Processing Units could be exploited for accelerating the rendering rates. Also, the current parallel computing methods and multi-core methods could be further used for achieving such acceleration. Third, with these ideas, it would be possible to teach/learn other subjects, such as animals/plants/etc. using different methods for classification. Changing these features is especially easy in our system because of its structure. The system could be used for other purposes and the results could be compared with the ones obtained in this work. Fourth, in order to evaluate the acquired knowledge of players, a final examination could also be included.

Now, we are developing new AR games for edutainment thanks to APRENDRA project. With it, we hope to contribute with new games, new devices that incorporate AR, new interfaces and validations with enough number of children for obtaining statistical significant results.

Finally, we firmly believe that AR has great potential in the educational field. Our results as well as those by other researchers (e. g. [Kau04]) should encourage the AR community to develop and evaluate new AR systems.

6. ACKNOWLEDGMENTS

We would like to thank:

- This work was partially funded by the APRENDRA project (TIN2009-14319-C02-01).
- The Summer School of the Technical University of Valencia for its collaboration.

7. REFERENCES

[Bil01] Billinghurst, M., Kato, H., Poupyrev, I. The Magic Book-Moving Seamlessly between Reality and Virtuality, IEEE Computer Graphics and Applications, 21 (3), 6-8, 2001.

[Bim01] Bimber, O., Fröhlich, D., Schmalstieg, D., Encarnaçao, L.M. The virtual Showcase, IEEE Computer Graphics & Applications, 21 (6), 48-55, 2001.

[Des10] Despina, M., Nectarios, P., Isabelle, C., Panagiotis. Z., Loukia L. H., Yiorgos C. Comparative Study of Interactive Systems in a Museum. EUROMED, 250-261, 2010.

[Fje07] Fjeld, M., Fredriksson, J., Ejdestig, M., Duca, F., Bötschi, K., Voegtli, B.M., Juchli, P. Tangible User Interface for Chemistry Education: Comparative Evaluation and Re-Design, CHI 2007, 805-808, 2007.

[Kat99] Kato, H., Billinghurst, M. Marker tracking and HMD calibration for a video-based augmented reality, 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR'99), 85-94, 1999, http://www.hitl.washington.edu/artoolkit.

[Kau04] Kaufmann, H. Geometry Education with Augmented Reality, PhD Dissertation thesis, Vienna University of Technology, 2004.

[Lar05] Larsen, L.B., Jensen, R.B., Jensen, K.L., Larsen, S. Development of an automatic pool trainer, Conference on Advances in Computer Entertainment Technology (ACE'05), 83-87, 2005.

[She02] Shelton, B.E., Hedley, N.R. Using Augmented Reality for Teaching earth-sun relationships to undergraduate geography students, 1st IEEE International Augmented Reality Toolkit Workshop, 8 pag., 2002 http://depts.washington.edu/pettt/papers/shelton-hedley-art02.pdf.

[Sla94] Slater, M., Usoh, M., Steed, A. Depth of presence in virtual environments. Presence: Teleoperators and Virtual Environments, 3, 130-144, 1994.

[Syk08] Sýkora D., Sedláček D., Riege K. Real-time Color Ball Tracking for Augmented Reality, 14th Eurographics Symposium on Virtual Environments (EGVE'08), 9-16, 2008

[Woo04] Woods, E., Billinghurst, M., Looser, J., Aldridge, G., Brown, D., Garrie, B., Nelles, C. Augmenting the science centre and museum experience, GRAPHITE, 230-236, 2004.

APPENDIX

Question ID	Questions
I1	How much do you know about the animals that are at risk of extinction and the causes for this?
I2	Please, indicate the value that best describes your opinion with respect to: "We should provide greater protection to animals that are at risk of extinction in order to prevent their extinction"
I3	Please, indicate to what extent you would be willing to support initiatives to protect animals that are at risk of extinction?

Table 3. Entry questionnaire

Question ID	Questions
AG1	***Engagement and fun*** I enjoyed playing this game.
AG2	This game has been fun
AG3	***Easy to use*** Has it been easy to play?
AG4	***Perceived value*** I think playing this game can help me to learn the animals that are at risk of extinction
AG5	I would like to play again because it is interesting for me
AG6	***Attitudes*** Please, indicate the value that best describes your opinion with respect to: "We should provide greater protection to animals that are at risk of extinction in order to prevent their extinction"
AG7	***Motivation to change*** Please, indicate to what extent you would be willing to support initiatives to protect animals that are at risk of extinction?
AG8	***Intention to change*** As a result of playing this game, I will talk with my friends and relatives about the animals that are at risk of extinction
AG9	As a result of playing this game, I will think more about the animals that are at risk of extinction and the causes for this

Table 4. Post questionnaire

Question ID	Questions
P1	Have you had the sensation of playing with pieces where images and videos appeared over them (AR system)?
P2	Were there moments during the game when you thought that the images over the pieces were real?

Table 5. Presence questionnaire

Question ID	Questions
AP1	How much have you learned about the animals that are at risk of extinction and the causes for this?
AP2	Which game did you like the most?
AP3	Why?
AP4	Add any comment about the experience

Table 6. Final questionnaire

Setting the Parameters of the LFT Shape Matching Algorithm

J.S.M. Vergeest, A. Kooijman, Y. Song
Delft University of Technology
Landbergstraat 15, NL-2628 CE Delft, The Netherlands
j.s.m.vergeest@tudelft.nl

ABSTRACT

The LFT algorithm (Large Fat Tetrahedron) is used to detect congruent subsets amongst unordered point sets and forms the kernel of a partial shape matching method. Although the method yields several advantages and is relatively efficient, its performance depends highly on the choice of various geometrical threshold parameters, as *e.g.*, for the length difference of two edges. We present an overview of the key parameters of the algorithm and their influence on the computation, a guide line to provide an initial value for the parameters and we propose an approach to their automatic adjustment.

Keywords

Scan view registration, partial shape matching, fat tetrahedron, threshold parameters

1. INTRODUCTION

The LFT algorithm (Large Fat Tetrahedron) was designed to detect approximately congruent tetrahedrons in two point sets [Vergeest 2010]. If such tetrahedrons are found, they might indicate the overlapping region of partially matching shapes. The assumption was that if a large fat tetrahedron with particular dimensions occurs in point set *A*, the probability that a congruent tetrahedron is found in point set *B* is small, unless both tetrahedrons reside in the overlap region of *A* and *B*. Thus, LFTs can serve as indicators of partial shape matching. The algorithm will be briefly described in Section 2.

One important application of partial shape matching is 3D scanning of physical objects. To construct a geometric model from a physical object, multiple scan views are taken, each consisting of range data, *i.e.* 3D points representing the object's surface. Since the orientation of the object relative to the scanning device is different for different scan views, the collection of points from all scan views do not as such represent the object's surface. First the points need to be aligned to each other, that is be transformed to a common coordinate system. The process of aligning the scan views is called scan view registration. From the aligned point sets the surface of the object can be reconstructed, either fully or partially, depending on the coverage of the scan views. If the surface can be fully reconstructed, it can be assumed to represent the boundary of a volume, or solid model. Then a solid model can be derived, which can serve as input to a CAD (Computer-Aided Design) system for further modeling and processing. Basing a design on an existing object or on reuse of precedent models is an important paradigm in some industries, such as industrial design engineering. Such a method can be successful only when even occasional users of scanning devices can easily operate the system. However, the registration task is, even nowadays, still an impeding factor. In practice the user could be a stylist who has manually created a clay model of a future household device. Whereas taking the scan views of the clay model is a commonsense task to him/her, the registration of view pairs is not. The scanning system's manufacturer normally offers an interactive software package, allowing the user to designate correspondences he/she observes amongst the scan views, as to provide a starting position for a shape matching algorithm, typically based on the ICP (Iterative Closest Point) method. The user has to aligned each scan view with the set of scan views aligned previously. Generally this way of operating

the scanner is perceived as slow and tedious, both by incidental users and by trained operators.

Several approaches have been reported to the problem of partial shape matching. From here on we assume that the input data consists of unordered point sets only. That is, we will not rely on preprocesses that generate surface meshes, nor on additional information such as color, texture or material properties of the scanned object. We focus on the kernel problem of matching two point sets. Most methods make use of geometric descriptors and/or feature points. The geometric descriptor can take many forms, including moments, FFT coefficients, spin images etc [Johnson 1999]. Defining a feature using integrated quantities rather than using derivatives reduces the influence of noise [Gelfland 2005]. Another approach to diminish sensitivity to noise and data outliers is taken by [Aiger 2008]. He collects sets of 4 planar points in each of the two point clouds. If a particular set from one point cloud is approximately congruent with one from the other point cloud, a candidate corresponding pair of 4-points sets is found. If the 4-points set is relatively wide, then the method is less sensitive to noise. We refer to [Gelfland 2005] and references therein for a more extended description of registration methods.

Our approach is inspired by the 4-points congruent sets as in [Aiger 2008]. We look for 4-points sets which define a large fat tetrahedron (LFT). The assumption is that the geometry of a large tetrahedron is relatively rare and therefore can serve to detect correspondences in the two point clouds. However, since true correspondence exists in the overlap region only, an upper bound must be set to the size of the tetrahedrons. Secondly, since the number of fat tetrahedrons in point sets can be very large, a straightforward comparison of two sets of tetrahedrons (each derived from one point cloud) would not be efficient. Our algorithm derives a limited number of fat tetrahedrons from one point cloud. Then each tetrahedron is tested for being approximately congruent to any point neighborhood in the other point set. In [Vergeest 2010] we have speculated about the advantages of the LFT algorithm compared to other strategies. However, lacking a benchmark platform we cannot demonstrate this. In the next section the LFT algorithm will be briefly described. In section 3 we present the influence of parameters on the computational performance of the method. Conclusions and recommendations are given in section 4.

2. THE LFT ALGORITHM

Let two point sets A and B be given, originating from sampling of a portion of the surface of a three-dimensional object. There may exist subsets $A' \subseteq A$ and $B' \subseteq B$ such that A' and B' are samples of the same subsurface of the object. A' and B' are then said to represent an overlap region of the samples.

Let a set of sets B_i be a partitioning of B defined as follows. A three-dimensional grid is constructed, aligned with a bounding box of B. The grid has the size of the bounding box of B. The block-shaped grid elements, or cells, all have the same size and have index i, $i = 1, ..., G$, where G is the number of cells of B. Each cell encloses zero or more points of B. Each point of B is enclosed by exactly one cell. B_i is the set of points enclosed by cell indexed i.

Let A' and B' be the largest overlap of A and B, informally defined as follows. Assuming that A and B are range images of a physical object, let S_A and S_B informally be defined as the portions of the surface of the physical object represented by A and B, respectively. Then, both A' and B' represent all or some part of the physical overlap surface $S_A \cap S_B$. Depending on the extent of $S_A \cap S_B$, A' and B' each may contain zero up to as many points as the cardinality of A and B, respectively.

Let $B'_i = B_i \cap B'$, that is the portion of cell B_i coinciding with the overlap region. Our search strategy is based on the assumption that the overlap region is connected and has the extent of at least the size of a cell. In such cases there might exist sets B_i containing multiple points of B_i'. A property of any point of B' is that its Euclidian distance to A is relatively small, provided that A and B are defined in the same coordinate system. However, since A and B originate from independent sampling processes, they will in general be defined in different coordinate systems. The difference between the two coordinate systems can be described by a rigid body transformation M, such that MB and A are defined in the same coordinate system, where MB is the set of points of B to which transformation M has been applied. We name this transformation the *matching transformation*.

Since neither the overlap region, nor the matching transformation M are known, we determine which of the sets B_i is fully or partly contained in B'. We do so by constructing the largest and fattest tetrahedron in each cell and test each such LFT against congruency with 4-points sets of A.

2.1 Finding the LFT

When a small set of points of B_i is close to A *and* if these points are sufficiently non-planar, then the transformation to match this set with A is a relatively good candidate of the M we are looking for. Relying on this principle we base the algorithm on matching 4 points to A, where the 4 points are contained in the same cell. The 4 points, denoted v_1, v_2, v_3, v_4, are

selected from B_i such that they form an LFT as follows: v_1 and v_2 are the points in B_i which are furthest apart. v_3 is the point in B_i furthest from the line through v_1 and v_2, that is it maximizes $|(v_3 - v_1) \times (v_2 - v_1)|$. v_4 is the point in B_i furthest from the plane defined by v_1, v_2 and v_3, that is it maximizes $|((v_3 - v_1) \times (v_2 - v_1)) \cdot (v_4 - v_1)|$. An example of an LFT is shown in Figure 1.

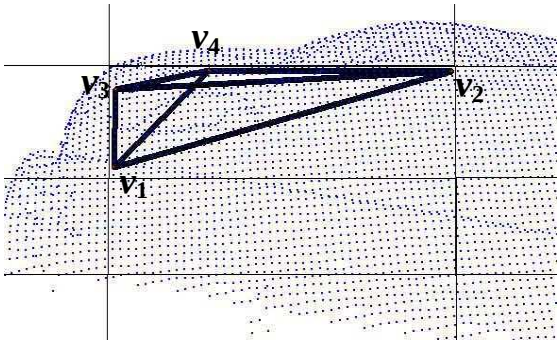


Figure 1. Point cloud *B*, its cell structure and the LFT contained in a cell. Data are from the Stanford Bunny [Stanford 2010].

2.2 Calculating the transformation *M*

Once an LFT has been determined in a particular cell of point cloud *B*, we look for potential corresponding 4-points sets in *A*. As mentioned, when the LFT resides in the overlap region of shapes *A* and *B* then there should exist 4 points in *A* representing a tetrahedron with dimensions equal to those of the LFT, that is up to some precision since the point sets *A* and *B* are obtained as independent measurements. The deviation between "corresponding" points can be expected to be as large as half of the scan spacing practiced. We have applied two methods to detect (approximately) congruent 4-points sets in *A*. We look for points a_1, a_2, a_3 and a_4 in *A* such that there exists a transformation *M* with $Mv_i \approx a_i$ for i=1,...4. The edge lengths of the LFT are denoted $l_{ij} = |v_i - v_j|$.

Method 1

For each point in *A*, name it a_1

Translate the LFT such that v_1=a_1

Search for points in *A* at distance l_{12} from a_1 and name them a_2

For each such a_2

Search for points in *A* at distance l_{13} from v_1 and at distance l_{23} from v_2 and name them a_3

For each such a_3

Rotate the LFT about v_1 such that v_2 gets closest to a_2

Rotate the LFT about axis (v_1, v_2) such that v_3 gets closets to a_3

If then v_4 is close to any point in *A* (called a_4) the accumulated transformations so far applied to the LFT represent a candidate *M*

End of method 1

Alternatively we can explicitly test congruency by comparing the six edge lenths of the LFT to the corresponding distances between candidate points a_1, a_2, a_3, a_4. We define δ_1 as the threshold value for length comparisons.

Method 2

For each point in *A*, name it a_1

For each point in *A*, name it a_2

If $|\text{dist}(a_1, a_2) - l_{12}| < \delta_1$

For each point in *A*, name it a_3

If $|\text{dist}(a_1, a_3) - l_{13}| < \delta_1$ and $|\text{dist}(a_2, a_3) - l_{23}| < \delta_1$

For each point in *A*, name it a_4

If $|\text{dist}(a_1, a_4) - l_{14}| < \delta_1$ and $|\text{dist}(a_2, a_4) - l_{24}| < \delta_1$ and $|\text{dist}(a_3, a_4) - l_{34}| < \delta_1$

Compute *M* from the v_i and a_i.

End of method 2

In method 2 the calculation of the transform *M* is postponed until the congruency is fully checked. One way to obtain *M* (as we implemented it) is to concatenate the translations and rotations in exactly the same way as done in method 1; see [Vergeest 2010] for the explicit equation. The two sets (v_1, v_2, v_3) and (a_1, a_2, a_3) are sufficient to determine *M*, but they do not lead to a set of linear equations with a unique solution since the congruency is approximate only. However, a solution based on minimizing $\Sigma |Mv_i - a_i|^2$ would be feasible and accurate (not implemented).

2.3 Computing the degree of overlap

Typically thousands of candidate *M* transforms are found, depending on threshold δ_1. If δ_1 is increased the number of candidates will rise steeply, as discussed later. We need to test whether or not a particular *M* is the matching transform. If an LFT *L* is contained in *B´* then the directed Hausdorff distance of *ML* to *A* will be (by definition) small if *M* is the matching transformation. It can be expected that then a significant portion of the points MB_i (from the current cell) will be close to *A* as well. Conversely, when many points MB_i appear close to *A* the probability that *M* is the matching transform is large.

In our algorithm, all points in cell B_i are subjected to M and their distance to A is determined. If a sufficient fraction f of the points are closer than δ_2 to A then M is saved as a candidate transformation. As a final step, each of the candidate transformations is used to compute the set MB, involving all points of B. The degree of matching of B to A is defined as N, the number of points in MB closer than δ_3 to A. The transformation producing the largest N is the outcome of the method.

3. PERFORMANCE AND PARAMETERS

As reported in [Vergeest 2010] the algorithm has been successfully applied to practical scan view registration. A typical CPU time of partial shape matching was 500s, which could be reduced to about 10s in a CUDA-GPU implementation [Kooijman 2009].

We have now studied the influence of the parameters δ_1, δ_2, δ_3 and f on the computational performance of the algorithm for method 2. The granularity of the subdivision into G cells is also of influence to the algorithm. Not all cells produce an acceptable LFT. We have set a lower limit to the number of points from B contained in a cell; if the cell contains too few points we do not consider it. If a particular LFT is too small or too thin, it is discarded. Therefore, typically 10% of the cells produce an LFT for further processing. We focused on method 2 since its implementation is relatively simple and it will be compared to its CUDA implementation in the near future. An upper bound of the complexity of the algorithm is $C \propto GP^6$, where P is the number of points occurring in a scan view (we assumed that A and B are of comparable size). In the search process, for each cell, each point in A is visited at least twice in order to form the line segment a_1a_2. When the test against l_{12} is passed, another loop over all elements of A is made to find candidates a_3 and finally one more loop to find a_4 (the maximal cost is proportional to GP^4 so far). If a_4 is found, then all points in the cell are compared to A (cost proportional to P^2) and possibly another check of all B against A is performed, as described in section 2.3. The main terms of the cost C are:

$$C \propto P^2 + \beta_1 P^3 + \beta_2 P^4 + (\beta_3 + \beta_4)P^6. \qquad (1)$$

β_1 is the probability that the l_{12} test is passed, β_2 is proportional to the probability that the l_{13} and l_{23} test are passed and β_3 is proportional to the probability that the l_{14}, l_{24} and l_{34} tests are passed. β_4 depends on parameter f and the degree of overlap found of points in the current cell with A.

If we would set δ_1=0 then no LFT would practically pass the l_{12} test and terms 2 and 3 would vanish, or $\beta_1 = \beta_2 = \beta_3 = \beta_4 = 0$. If aforementioned fraction f and all δ_i are large then C behaves like an 6th power function of P for large P; the number of candidate M transforms would be very large and many of them have to be checked by Hausdorff twice, namely once involving MB_i and possibly once more involving MB.

To achieve efficient partial shape matching the algorithm should detect the matching M (therefore, the parameters should not be too small), without superfluous candidates (therefore, the parameters should not be too large).

To gain insight in the effects of the parameters we have performed numerical tests on one particular set of A and B, with cardinality 3188 and 2407, respectively. These sets are down-sampled versions of the Bunny data from the Stanford Scan Data Repository [Stanford 2010]. The data points are relatively evenly spaced, about 2mm apart, on a surface with a diameter of approximately 150mm.

Table 1 gives an impression of the computational expense of the algorithm for Method 2. The 10 runs differ only by the parameter δ_1. The cell division was $G = 5\times5\times5 = 125$ equally sized blocks of 24×31×27mm. Out of these, 77 were empty and 7 cells contained an LFT that was sufficiently large and fat. Out of these 7, only one LFT appeared to be located in the overlap region and could produce the correct matching transform. This particular LFT had l_{12} = 40.6mm and a base triangle of height 20.2mm (that is the distance of point v_3 to edge v_1v_2) and thickness 3.1mm (distance of v_4 to the base). The algorithm includes threshold parameters for thickness to accept LFTs and also for the minimum number of points contained in cells that are considered as carrier of an LFT. For the runs of Table 1, the limitation parameters were chosen 3.0mm for height and 96 for the number of points in a cell. The latter number was calculated as $n_B / G^{2/3} = 2407/25 \cong 96$, which reflects the fact that the data points represent a dimensionality 2 boundary rather than a volumetric content. The cell located in the overlap region contained 100 points. We have set fraction parameter f = 0.9. If 90% or more of the cell points after transformation got closer than δ_2 to any point of A then the LFT yielded "cell OK" in the table. δ_2 was set to 1.5mm. The threshold for computing the overlap explicitly was set to δ_3 = 1.5mm. When the correct overlap (and thus the correct matching transform) was found, the associated LFT was classified as "all OK".

From the table we see that δ_1 = 0.67 is approximately the lowest threshold for which at least one of the 7 LFTs produced the correct matching transform M. The number of points from MB closer than δ_3 = 1.5mm to A was 540 (22.4%), reflecting the size of the overlap region (we were able to judge the

n_A = 3188, n_B = 2407, G = 125, LFTs = 7, δ_2 = 1.5mm, δ_3 = 1.5mm, f = 0.9									
δ_1 (mm)	a_{12}	a_{12} OK	a_{123}	a_{123} OK	a_{1234}	a_{1234} OK	cell OK	all OK	CPU(s)
0.00	7.1×10^7	0	0	0	0	0	0	0	2
0.15	7.1×10^7	2.0×10^5	6.3×10^8	6.5×10^3	2.0×10^7	10	0	0	18
0.30	7.1×10^7	4.0×10^5	1.3×10^9	5.1×10^4	1.7×10^8	438	0	0	40
0.45	7.1×10^7	5.9×10^5	1.9×10^9	1.8×10^5	5.6×10^8	4345	2	0	95
0.60	7.1×10^7	7.9×10^5	2.5×10^9	4.2×10^5	1.3×10^9	22,900	4	0	330
0.66	7.1×10^7	8.7×10^5	2.8×10^9	5.6×10^5	1.8×10^9	36,638	6	0	410
0.67	7.1×10^7	8.9×10^5	2.8×10^9	6.9×10^5	1.9×10^9	45,274	10	1	460
0.75	7.1×10^7	9.9×10^5	3.2×10^9	8.2×10^5	2.6×10^9	83,868	15	2	780
1.05	7.1×10^7	1.4×10^6	4.4×10^9	2.3×10^6	7.2×10^9	5.8×10^5	64	4	4800
1.50	7.1×10^7	2.0×10^6	6.3×10^9	6.6×10^6	2.1×10^{10}	4.5×10^6	371	26	36,700

Table 1. Complexity terms of length tests by Method 2 as function of threshold δ_1. Data are from the Stanford Bunny.

correctness of the transformation of these particular scan views based on ground knowledge from alternative shape matching methods). Out of the 100 points in the particular cell containing the "correct" LFT, 97 were at close distance to *A* (within threshold δ_2 = 1.5mm).

Obviously, the 4 vertices $v_1...v_4$ of the LFT itself belong to that set of 97 points, since they are close to the points $a_1...a_4$. For the same LFT one different 4-points set in *A* was found leading to a transformation passing "Cell OK". With that transformation 91 points of the cell were close to *A* but so were only 159 points of *B*, making it very unlikely that the transformation was the matching transform. Another LFT generated 8 transforms passing "Cell OK" but turned out not the matching transform. The total of 10 cases of "Cell OK" is depicted in Figure 2. The number of overlap points in *B* is plotted against the rotation exerted by the transform. The latter quantity was chosen as it characterizes the transform, although there is, of course, not a one-to-one relationship between the angle and the transformation matrix. The orientation of scan view *B* relative to *A* is 89.9 degrees, according to the solution found at δ_1 = 0.67. The number of l_{12} tests is, independently of δ_1, equal to n_A^2 times the number of LFTs considered. For the runs of Table 1 this number, labeled a_{12}, was $3188^2\times7 = 7.1\times10^7$. For δ_1 = 0.67, 1.3% of the l_{12} comparisons passed the test, labeled "a_{12} OK". This percentage is collective over all accepted LFTs in the run. The number of (l_{13}, l_{23}) tests is n_A times the "a_{12} OK" cases, or 2.8×10^9. This number depends on the third power of n_A. Further, β_1 in equation (1) is dependent on δ_1.

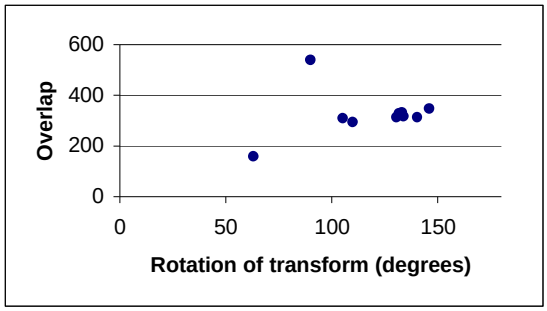


Figure 2. Number of points in *MB* close to *A* versus rotation of *M* for δ_1 = 0.67 and *f*=0.9. The plot contains 10 data points.

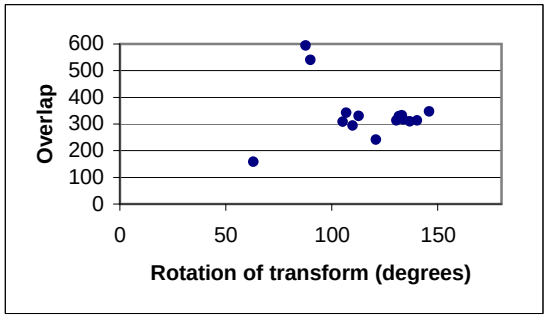


Figure 3. Number of points in *MB* close to *A* versus rotation of *M* for δ_1 = 0.75 and *f*=0.9. The plot contains 15 data points.

For the particular runs in Table 1 it turned out that β_1 = 0 for $\delta_1 \le 0.66$ and β_1 is rising for increasing δ_1. For δ_1 = 0.67, 0.02% of the a_{123} tests was positive. This percentage is proportional to β_2 and depends on δ_1. From the 1.9×10^9 a_{1234} tests, 0.002% or 45,274 resulted positively. This fraction affects β_3. Finally, the number of exhaustive tests of distance *MB* to *A* depends on the fraction of "cell OK" (0.02% in the run for δ_1 = 0.67). This fraction is proportional to β_4.

The increase of computation time with increasing δ_1 is obvious from the rightmost column of Table 1. The choice of δ_1 seems most critical, whereas δ_2 and f affect the number of distance evaluations, which will be increasingly critical for large n_A and n_B.

The effect of changing f from 0.9 to 0.8 is shown in Figure 5, which should be compared to Figure 3. The number of candidates passing the f threshold rises from 15 to 308. Instead of having only two correct transformations for f=0.9, there are 4 of them for f=0.8. They show up as a narrow peak in Figure 5 near 90°.

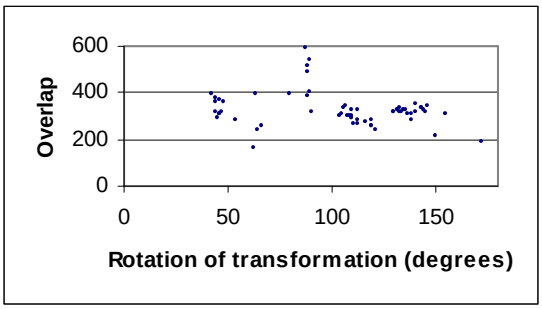


Figure 4. Number of points in *MB* close to *A* versus rotation of *M* for δ_1 = 1.05 and f = 0.9. The plot contains 64 data points.

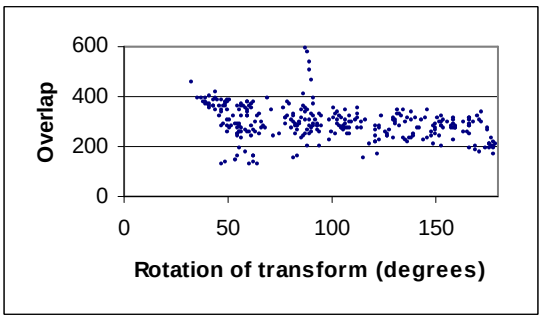


Figure 5. Number of points in *MB* close to *A* versus rotation of *M* for δ_1 = 0.75 and f = 0.8. The plot contains 308 data points.

Lowering f to 0.5 does not lead to more correct transforms, but increases the background of incorrect candidates (Figure 6). In Table 2 the number of candidates and the performance of the algorithm, for the different values of f, are presented. It should be noted that coefficient β_4 increases with decreasing f, leading to 6th power behavior of the complexity, see equation (1). The increase of CPU time in Table 2 can be practically completely attributed to the number of full distance computations. When the number of tested cells increases from 308 to 8695 (factor 28.2), the CPU time for *B*-to-*A* distance computation increases by factor (2114-780) / (829-780) = 27.2. In all cases the number of cell-to-*A* distance computations is 83,868, which would increase with the 6th power of number points as well. However, the β_3 coefficient is small when δ_1 is moderate and the number of points in a cell is small.

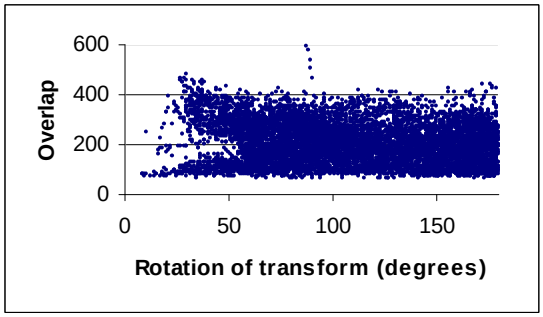


Figure 6. Number of points in *MB* close to *A* versus rotation of *M* for δ_1 = 0.75 and f = 0.5. The plot contains 8695 data points.

n_A = 3188, n_B = 2407, G = 125, LFTs = 7, δ_1 = 0.75mm, δ_2 = 1.5mm, δ_3 = 1.5mm				
f	a_{1234} OK	cell OK	all OK	CPU(s)
0.9	83,868	15	2	780
0.8	83,868	308	4	829
0.5	83,868	8695	4	2114

Table 2. Dependence on f of the performance of Method 2.

We observed that increasing δ_1 and/or decreasing f puts a dramatic load on the computation. Further, having approximately 1.6 times more unordered points in the sets, rises the CPU times by a factor 20.8, see Table 3 for details. For a pure 6th order behavior one would expect a factor 16.8, but there are other factors such as the number of accepted LFTs, which changed from 7 to 9.

4. CONCLUSIONS

The two most critical factors influencing the performance of the algorithm are the point set sizes and δ_1. In the test runs we have used down-sampled versions of point sets containing originally 40,200 and 30,400 points. The down-sampling algorithm removed points conservatively from the set such that no two points were less than ε length units apart, where ε was set to 2mm for the runs in Tables 1 and 2, and ε = 1.5mm for the runs in Table 3. For given ε an upper limit of δ_1 would be 0.5ε in a worst-case one-dimensional setting. Using a small value for δ_1 could lead to zero LFT matches.

n_A = 3188, n_B = 2407, G = 125, δ_1 = 0.75mm, δ_2 = 1.5mm, δ_3 = 1.5mm, f = 0.9									
n_A, n_B, LFTs	a_{12}	a_{12} OK	a_{123}	a_{123} OK	a_{1234}	a_{1234} OK	cell OK	all OK	CPU(s)
3188, 2407, 7	7.1×10^{7}	9.9×10^{5}	3.2×10^{9}	8.2×10^{5}	2.6×10^{9}	83,868	15	2	780
5140, 4127, 9	2.4×10^{8}	3.1×10^{6}	1.6×10^{10}	4.0×10^{6}	2.1×10^{10}	737,044	703	40	16,240

Table 3. Effect of increasing the density of the point sets by approximately a factor 1.6 for Method 2.

Empirically we have found δ_1=0.66mm=0.33ε as an upper limit in the particular setting of the runs we performed. This could be considered as a rule of thumb for δ_1, although it presumes evenly spaced points. The choice of G has turned out to be critical as well. When we selected G=6×6×6=216, none of the LFTs yielded a correct transform, unless we increased δ_1 from 0.67 to 1.05mm. Indeed, refining the cell subdivision could exceptionally imply that *fewer* cells are fully included in the overlap region. When we lowered G to 4×4×4=64, none of the LFTs yielded a correct transform, not even at δ_1 = 1.05. This could have been expected since the LFTs all exceed the size of the overlap region and are therefore unlikely to fit "correctly" to A at any place.

The subdivision has been implemented on an arbitrarily orientated evenly spaced grid, namely a grid aligned with the global coordinate system. This subdivision method could be improved significantly to reduce the number cells that should be considered further. If we assume that the overlap region contains at least 20% of the points of B then the 5×5×5 subdivision seems appropriate.

The values of δ_3 and δ_4 are less critical, *provided* that δ_1 is not unnecessarily large. The values we supplied (1.5mm or 0.75ε) seem reasonable. f=0.9 turned also a good starting value; the correct transforms yielded about 95% overlap of the cell with A, and we observed that even with increased δ_1, it was not useful to set f lower than 0.9.

As mentioned, these recommendations for initial parameter values are still case-specific. When the scanning process would result in very unevenly distributed points, the parameters should be derived from the highest occurring ε.

A possible strategy for automatically adjusting the parameters is to set ε relatively large (*e.g.* 0.01 times the diagonal of the bounding box of A) and perform a run with δ_1 = 0.3ε. Then the largest overlap detected can be tested for compactness and connectedness. If the distribution of the overlapping points is not consistent with a connected portion of A and/or B, runs with increased δ_1 can be carried out.

Both the length tests and the distance computations can be implemented with a good degree of parallelization. Unlike purely two-dimensional processes such as image restoration, 3D scan view data cannot be subdivided in portions which can be processed completely independently. Still an acceleration of the computation by a factor of 10 to 100 appears feasible in Cuda implementations that are presently under investigation.

REFERENCES

[Aiger 2008] Aiger, D., Niloy, M., Cohen–Or, D. 2008. 4–Points Congruent Sets for Robust Pairwise Surface Registration. ACM Trans. Graph. 27, 3.

[Johnson 1999] Johnson, A.E. and M. Hebert, "Using spin-images for efficient multiple model recognition in cluttered 3-D scenes," IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 5, pp. 433–449, 1999.

[Gelfland 2005] Gelfland, N., Mitra, N. J., Guibas, L. J., Pottmann, Robust global registration. In Proc. Symp. Geometry Processing, Euro-graphics, pp 197–206.

[Kooijman 2009] Kooijman, A., Vergeest, J.S.M., A GPU-supported approach to the partial registration of 3D scan data. Proceedings of the Gravisma 2009, D. Hildenbrand and V. Scala (Eds), pp 1-5.

[Stanford 2010] Stanford Scan data Repository, http://graphics.stanford.edu/data/3Dscanrep

[Vergeest 2010] Vergeest, J.S.M., Kooijman, A., Song, Y., Partial 3D shape matching using large fat tetrahedrons. Journal of WSCG, Vol. 18, Nr. 2, pp 41-47, 2010.

Plausible Visualization of the Dynamic Digital Factory with Massive Amounts of Lights

Franz Peschel
Daimler AG
franz.peschel@daimler.com

Fabian Scheer
Daimler AG
fabian.scheer@daimler.com

ABSTRACT

In the last years an enormous progress has been made in improving the visual quality of virtual worlds through approximations of former computationally expensive global illumination (GI) effects. Nevertheless, the handling of dynamic or even deformable content in massive data sets illuminated and shadowed by massive light sources still represents some kind of killer application. Besides the gaming industry, these demands are especially made in digital factory planning applications, where moreover preprocessing times and additional manual preparations of the virtual scenarios have to be avoided to keep a short time to production and accordingly to market. Therefore, we present an efficient rendering pipeline and concept for the creation of a plausible illumination for the digital factory, able to handle dynamic content in massive data sets at a large extent, that are plausibly illuminated and shadowed with a lot of light sources and encoded with high dynamic range information at interactive frame rates in a high resolution. The most important aspects for a visually plausible result are analysed on basis of real images of a factory and at last evaluated with the achieved results. Finally, we give a detailed overview and analysis of the performance of the incorporated techniques on modern graphics hardware to identify the main bottlenecks and key points for future research and conclude with an extensive reflection of the benefits of a plausible illumination for digital factory planning applications.

Keywords: Global illumination, massive data, massive light sources, plausible visualization, virtual reality, digital factory

1 INTRODUCTION

Nowadays, the creation of digital mock-ups (DMUs) in early stages of the product or production life cycle is an essential planning instrument to increase the quality and reduce the costs of products or production facilities. Possible bottlenecks or sources of errors can be discussed and analyzed by experts on basis of a holistic visualization. Nevertheless, an entire factory DMU is very complex (e.g. $>$ 25 mio. polygons), consisting of a lot of data contributed by several planning departments or different suppliers. For this reason special rendering methods, like the visibility guided rendering (VGR) described in [12], have to be applied to handle these massive data sets at interactive or realtime frame rates. In this context the visualization plays an important role. In the past, real time or interactive GI effects for such massive amounts of data were hard to achieve, so expensive cluster solutions were applied to obtain a realistic impression. Furthermore, simple flat or phong shading was the only way to achieve the interactivity with the virtual world in real time frame rates. As a consequence stereoscopic techniques in combination with powerwall systems had to be incorporated, to support the perception of depth and the correct assessment of the digital content, like object correlations. This benefit predominantly arised through the binocular disparity depth cue [3], but also neglected an improved effect for the users perception or feeling of being immersed when supporting a better representation for the perception of pictorial depth cues [3, 17], which are evoked by a realistic visualization. Due to the evolved capabilities of modern graphics hardware former computationally expensive GI effects can be approximated at interactive or real time rates. A high visual quality of virtual worlds can be achieved on a single desktop PC, paving the way for desktop virtual reality(VR) and enabling the possibility to reduce the costs considering the provision of expensive hardware.

Against this background we extend former work[19, 23] and present an efficient rendering pipeline and concept for a plausible visualization of the digital factory (DIFA). Dynamic and deformable content of massive data sets illuminated and shadowed by massive amounts of light sources can be handled. Various GI effects are achieved by approximation, but produce a convincing result. The most important and necessary factors of a plausible visualization of the DIFA are analyzed on basis of footages of a real factory and compared with the achieved quality of our results. At the end a detailed overview of the performance of the several techniques to achieve the identified important factors is given, serving as a basis for the identification of the main bottlenecks and key points of future research in this field of application. Our main contributions are the analysis of the key points for an approximated realistic visualization of the digital factory, the specified efficient rendering pipeline for the DIFA and the detailed performance evaluation of the pipeline on modern graphics hardware.

2 RELATED WORK

A lot of progress has been made in the last decades to improve the realism of virtual worlds by global illumination(GI) algorithms at interactive or real time frame rates. [6, 1] present interactive ray tracing to account for massive data sets in virtual reality(VR) applications. [29] and [18] adopt the capabilities of modern graphics hardware and push ray tracing to the next level by considering more complex light effects[29]. Due to the logarithmic complexity, ray tracing is best suited to handle large data models at performant frame rates. Nevertheless, the limits are revealed when dynamic massive data sets illuminated by massive amounts of light sources and the expensive computation of secondary effects, like bounces of indirect lighting, are involved.

More practical approaches for a dynamic and interactive plausible illumination of complex and massive scenes with a lot of light sources can mainly be found by having a look at the gaming industry([25, 5, 16, 15, 26]), especially when desktop VR with standard hardware is desired to avoid expensive hardware solutions. In this context big advancements were made through image space approximations and deferred shading techniques in the last years. [25] uses deferred shading, introduced by [2], to consider huge amounts of lights for shading and relies on visible fragments only. Thus, the computational effort mainly depends on the amount of light sources and the image resolution, while simultaneously being independent of the polygon count and allowing the handling of dynamic or deformable content in real time.

[4] further improves the performance of this method by accumulating the illumination per pixel in a separate light accumulation buffer(LAB) without the consideration of the material color, which is multiplied at the end, thus reducing computational efforts when a lot of light sources are considered. However, occlusion information is missing and the creation of shadow maps for many light sources would still represent the bottleneck. In addition to the consideration of direct light effects([27, 7]), many advancements have been made for approximating GI effects at interactive or real time rates, like screen space ambient occlusion (SSAO)[16], which has become a standard in the gaming industry. In [23] vector based SSAO closer to the original definition of ambient occlusion is presented, especially focusing on the application in digital factory planning scenarios. In [22] SSAO is extended to account for directional occlusion and indirect lighting effects by screen space directional occlusion(SSDO), but is limited to nearly arranged surfaces. In the following sections the indirect lighting effects of SSDO are also mentioned as screen space global illumination(SSGI). [21] present an interesting approach considering the shadow computation of indirect illumination effects approximated by the usage of many secondary light sources. Unfortunately, their approach relies on a point based scene representation, introducing a further preprocessing step we want to avoid to keep minimal production times. In summary, screen space approaches work entirely in screen space without the need for preprocessing steps while simultaneously being independent of the scene complexity.

Recently, [11] introduced cascaded light propagation volumes (CLPVs). In contrast to SSAO or SSGI, CLPVs can compute indirect lighting effects for large areas and are not limited to reveal its effect in cavities or creases only. Three different sized cascaded grids are attached to the camera and indirect illumination is propagated through the grids by using spherical harmonic functions, yielding a performant GI computation. However, small areas in the scene can be missed due to the grid resolution, but fortunately reconsidered with SSGI.

3 SOLVING THE RENDERING EQUATION

According to [9] the rendering equation can expressed as a *Neumann series* using the integral operator notation in the following form[8]:

$$L = E + KL, \tag{1}$$

where K represents the integration over the cosinus modulated BRDF. E denotes the emitted radiance L_e and L the reflected radiance L_r. Equation (1) can be discretized to a simple matrix equation, where E and L are vectors and K is the light transport matrix, characterizing the reflectivity of light in a scene, yielding:

$$L = E + KE + K^2E + K^3E + ... = \sum_{m=0}^{\infty} K^m E \tag{2}$$

E denotes the emitted radiance of a light source, KE the direct illumination of surfaces, K^2E the first bounce indirect illumination, K^3E the second bounce and so on, finally converging against the exact solution of the rendering equation. On basis of this representation the following methods for the computation of single stages of our rendering pipeline and the analysis of important effects in images of a real factory can be broad into a better coherence. Thereby, the operator notation underlines the separability of GI into single computation stages. To achieve an interactive performance of our pipeline we restrict ourself to consider equation (2) just to K^2E, which in most cases is sufficient for a plausible illumination simulation, since the perceptually noticeable differences between the summands strongly decrease from the fourth summand.

4 ANALYZING THE REAL FACTORY

To evaluate the plausibility of an illumination simulation, comparisons to the reality in the factory or at least to images of the real factory have to be done. Furthermore, essential illumination effects for a plausible visualization can be identified by analyzing these images. Several images were used for the analysis under consideration of the summands of the discretized rendering equation (2) in section 3. The four most representative images are presented in fig.2. In addition, various effects influencing the perception of the human visual system are also addressed in the analysis.

4.1 Emission - E

Referring to equation(2), E represents the emitting radiance of light sources. This effect can be noticed in the views of fig.2 at windows with incoming sunlight or at the fluorescent tubes mounted near the ceilings. Since the images were taken with a low dynamic range camera light emitting surfaces appear as bright white areas, thus no further illumination effects can be recognized in this areas. Factories are mainly illuminated by fluorescent tubes and sunlight that shines through windows and in conclusion have to be simulated in a realistic way, since the human perception strongly reacts to these high frequent parts of the illumination.

4.2 Direct Illumination - KE

In absence of strong sunlight on a sunny day and deep inside the buildings without any windows in range, factories are homogeneously illuminated to achieve ergonomically illuminated workplaces. As a consequence, a lack of hard shadows can be noticed, except for areas under machines and obstacles that are mounted some inches above the ground. The only source for hard shadows represents the sun light, when falling through the windows in a strong intensity. In conclusion, shadow mapping techniques should be applied for the sun light, whereas a diffuse ambient occlusion is sufficient concerning the weak occlusion caused by the fluorescent tubes. Since the tubes are encapsulated by a socket their emitting hemisphere is directed towards the ground.
To achieve a plausible illumination the most remarkable materials also have to be considered to create a visually convincing result, as can be seen by the several specular or glossy effects on the ground and at the pipes in fig.2. The reflecting ground can especially be noticed in view 1 and view 4 and the glossy pipes in view 2 and view 3. Furthermore, strong diffuse reflections can be recognized in view 2 at the walls or at the machinery.

4.3 Indirect Illumination - K^2E

The homogeneous distribution of the light in the scenes as well as the strongly diffuse reflecting surfaces create a smooth ambient illumination and thus it evokes the effect of ambient occlusion in corners and interspaces. Therefore, the ambient brightness and occlusion has to be simulated to obtain a plausible result. Due to indirect bounces of the light in the factory all images of fig.2 seem to be tinted with a green shimmer, which is mainly caused by the green machinery (see view 2 and view 4). As a consequence, the incorporation of indirect illumination effects is essential to achieve a similar appearance concerning indirect bounces and the ambient brightness.

4.4 Glare

Due to the limited capabilities of the used low dynamic range camera used for the images of the factory, the surroundings of light emitting areas are overexposed and glow effects can be recognized at the outer regions. The same effect can be seen at strong reflections on the ground or at metallic surfaces. Those effects are also perceived by the human visual system, even though in a reduced form since the eye can adapt to such illumination situations to a certain degree.

5 DESIGN OF A MASSIVE LIGHTING PIPELINE

In this section a short overview of the massive lighting pipeline (see fig.1) is given, specifying how the several methods for a plausible illumination computation are incorporated into the rendering module of a VR system. For the implementation of the massive lighting pipeline we focused on the visualization of the interior of the factory. As a consequence, hard shadows inside the buildings that merely occur due to strong sun illumination on a sunny day(see section4) are not considered, but can additionally be incorporated as described in our former work[21]. Due to the homogeneous illumination inside the facility convincing occlusion results can be obtained by the SSAO approach of [25]. The main focus lies on the plausible dynamic illumination by thousands of light sources in massive data sets and the simulation of diffuse, glossy and reflective materials. Indirect illumination effects in the nearfield of cavities or creases are achieved by applying SSGI and for larger areas by the application of CLPVs. An ambient term is added by exploiting the capabilities of CLPVs to directly propagate direct light information, which is physically incorrect, but yields visually convincing results, as can be seen in view 2 in fig.2. As a consequence, massive data sets illuminated with massive lights and containing dynamic or deformable content can be handled at interactive rates and in high resolutions without the need for preprocessing stages.

5.1 GBuffer Layout

Since factory planning scenarios with a large spatial extent are considered, the position and normal informa-

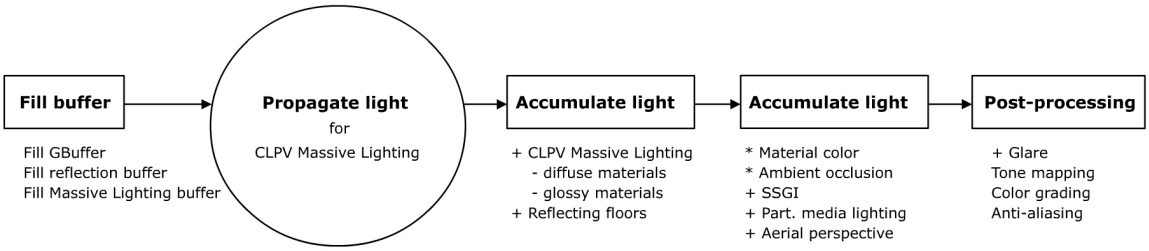


Figure 1: Overview of the massive lighting rendering pipeline

tion has to be stored with 16-bit precision to avoid disturbing z-fighting artifacts as well as gradation artifacts of surface shading. For high screen resolutions this can lead to a large memory requirement. To reduce memory consumption, the world positions are reconstructed from one 16-bit depth value. For the normalized surface normals only the x and y components are needed, since the z component can be reconstructed too. For depth and normal we use the RGB channels of a first render target RT1. Additionally the material color is stored in the RGB channels of a second render target RT2. Further information like specular strength, transparency flags and proxy geometry flags are stored in the alpha channels of RT1 and RT2. In summation two RGBA floating point textures with 16-bit per channel are used for the geometry buffer(GBuffer).

5.2 Massive Lighting

[10] describes the possibility to apply CLPVs for a direct diffuse illumination with massive light sources by directly injecting the direct light information into grids. Therefore, a light source buffer, further denoted as massive lighting map(MLM), is created that encapsulates a maximum of $512x512 = 262144$ usable light sources. The MLM contains position, color, orientation and intensity of light sources. Due to the small computational effort of the injection stage of the light sources into the grids, also higher resolutions of the MLM are possible. Hence, area lights like the fluorescent tubes can be properly approximated by many point lights since the amount of light sources will not become a bottleneck. Dividing the amount of all tubes in the factory by the MLM buffer resolution yields the number of possible light sources to approximate a tube. In fact, such a dense sampling of the tubes is necessary, since flickering artifacts can occur when a light source enters or leaves the spatial grid that is attached to the camera due to camera movements. By using a lot of light sources this case can be avoided due to a smooth fading between the grid cells. In the fill buffer stage of our pipeline the lights of the MLM are injected into the CLVP and propagated through the several grids in the propagation stage. In the light accumulation stage the propagated direct illumination can be accessed at the respective position in the scene to fill the LAB with the direct illumination.

5.3 Ground Reflections

To account for visible important details of reflections on the ground, the scene is rendered a second time into another frame buffer from the perspective of a mirrored camera to obtain a mirrored image. In the light accumulation step, the ground plate is automatically identified by a flag in the alpha channel of RT2 in the GBuffer, which is set using the material of the ground plate. Thus, the covered pixels in image space are determined and used as mask for the rendering of the ground reflections. For the simulation of surface roughness the reflections are finally processed with several filtering techniques, e.g. a light streaks approach described in [13] or a simple 5x5 gaussian blur filter.

5.4 Screen Space Ambient Occlusion and Global Illumination

Ambient occlusion simulates the occlusion in cavities, creases and for concave surfaces and is best suited to provide shadow information for uniformly illuminated diffuse or ambient environments[14], typical for factory interiors. Therefore, [23] is applied yielding sufficient results with merely eight to 16 samples in this diffuse environment. By means of the indirect lighting part of the SSDO technique, indirect bounces of light can be computed for nearly arranged surfaces. Since the method is applied to the LAB that already contains the direct illumination of the CLPVs at this stage, indirect bounces of the illumination can be computed as described in [19]. Due to the use of many light sources we do not apply the directional occlusion part of SSDO.

5.5 Participating Media Lighting

On basis of the CLPVs a spatial representation of the illumination is available, that can be further extended by the incorporation of direct light information from the fluorescent tubes to simulate participating media lighting effects. Light sources of the MLM are injected into the CLPV with a higher weighting factor and propagated. By using view aligned volume slicing with screen aligned quads, the light intensities of the CLPV grid voxels can be computed and added to the LAB, simulating the effect of homogeneous participating media lighting. By choosing an appropriate density factor various effects like fog, dust or aerial perspective can be

Figure 2: Top row: Flat shading. Middle row: Representative views for the analysis of a real factory, ranging from view 1(left) to view 4(right). Bottom row: Results of our plausible illumination

simulated. For scenes with a large extent this can also evoke the impression of an aerial perspective and thus increase the depth perception of the scene.

5.6 Tone Mapping, Glare and Color Grading

To account for a more realistic representation of the scene, the illumination computation is based on high dynamic range(HDR) information. Thereby, the capability of the human visual system to adapt to different lighting conditions is simulated by applying the tone mapping operator of [20], allowing a plausible display of the HDR information on low dynamic range output devices. The human perception of bright areas in the factories (see section 4) is also considered by using the approach of [13] to visualize a glare effect in the area of light emitting or strong reflective surfaces. Therefore, we store bright areas above a certain threshold in an additional image which is downsampled and stored multiple times by the factor four. The respective images are filtered with a 5x5 gaussian mask. Afterwards the images are upsampled again and added to the final image composition. Since physically correct indirect bounces of the illumination are neglected, the color temperature of the factory was simulated by a color grading method as it is described in [24] using a 3D Look-Up Table(3D LUT) which simply represents a mapping of a rgb color cube into another rgb color cube and allows us to do the color correction very fast. The color grading can be changed in run time using manipulation methods of the 3D LUT. It is also possible to store or load a 3D LUT.

5.7 Anti-aliasing

Aliasing effects caused by an undersampling of high frequent areas can lead to flickering artifacts and jaggy edges, especially during camera movements, reducing the feeling of being immersed in the virtual world. Because of the used deferred shading approach no hardware anti-aliasing can be utilized. We distinguish between a performance and quality version of anti-aliasing. For the performance version an edge anti-aliasing(EdgeAA) is implemented. Edges are detected by building the differences of the surface normals and depth values, followed by a 3x3 gaussian blur applied at the edges. The quality version uses full screen anti-aliasing(FSAA). Therefor the image is computed in doubled resolution. For the super sampling part of FSAA a rotated grid sampling pattern is applied, yielding an optimal result for nearly horizontal or vertical edges, which are often seen in factory environments.

6 RESULTS

To achieve an interactive plausible illumination of massive data sets with massive lights for digital factory planning applications the following main criterias are determined:

- Visually plausible quality
- Interactive or real time frame rates
- Support for massive data sets with dynamic content

- Support for massive dynamic light sources
- Avoidance of preprocessing time

In this section the achieved results are evaluated and reflected considering the listed criterias. The test system consisted of an AMD Athlon 3700+ with 2 GB RAM and a NVIDIA GeForce GTX 285 graphics card. For the performance measurements a typical factory scene with 15.3 million polygons and 4056 fluorescent tubes is used and rendered in a resolution of 1280x1024 pixels. Each of the 4056 tubes of the scenario is approximated by 5 light source, yielding a total of 20280 light sources. To handle the massive amounts of data efficiently a visibility guided renderer[12] is used as the rendering core of the VR-system, filling the buffers for our further illumination computations.

6.1 Visual Quality and Plausibility

To evaluate the plausibility of the proposed method, the images of the real factory and the criterias that were derived by the analysis in section 4 are compared with the achieved results. The results, ranging from view 1 to view 4 in the bottom row of fig.2, show that the light emitting fluorescent tubes with their surrounding glow are plausibly visualized[1]. However, the glow in the images seems a bit more smoothed at the outer edges and is sometimes colorful pigmented against dark backgrounds (see view 1 in fig.2), allowing a further improvement of the simulation. Even though, the direct illumination is spatially approximated by the CLPVs with two band spherical harmonics, the results are visually convincing. Nevertheless, artifacts in form of some isolated bright areas (see view 2 in fig.2) can occur due to the spatial approximation of the grids. These are mainly caused by high intensities in grid cells including direct light sources.

Direct light is propagated in a specific direction in form of cosine-lobes. Thereby, some amount is also send in the opposite direction yielding a scattering of the direct light in the scene. Because of the density of fluorescent tubes and multiple reflections of their light in the factory a similar effect of a diffuse distributed ambient environment can be observed in the images. As a consequence the massive lighting also simulates some kind of physically incorrect indirect illumination in the surroundings of light emitting surfaces and decreases with distance. Thus, a kind of global ambient term is obtained that decreases in distance to a light source in contrast a constant ambient factor, e.g. used in the OpenGL lighting model. However, despite the fact that the light propagation in form of cosine-lobes is directed to the ground, the areas above the fluorescent tubes are also illuminated due to the scattering, thus creating an indirect illumination that appears very similar to the images(see view 2 and view 3 in fig.2). Unfortunately, this effect is colored with direct light information and can not consider the color of a true bounce of indirect illumination. The provided occlusion information obtained by SSAO yields visually plausible results in most of the cases. The limits are revealed in view 2 in fig.2. The shadow beneath the machine in front should be stronger and weaker at the pillar. In view 1 in fig.2 the occlusion on the right side also seems to be to strong, but can be improved with a better parameterization. The indirect illumination effect of SSGI can be clearly noticed in view 2 in fig.2 on the left side on the ground near the green machines, further improving the visual quality towards a plausible impression.
Another important aspect concerning the plausibility is the reflection of the scene on the ground, even though it sometimes appears to be polished due to a lack of the simulation of rough surfaces, which can be incorporated by using additional bump and specular maps[2]. The sparse application of participating media effects in the background of the scenes decreases the contrasts in distance and creates an aerial perspective that further conveys the impression of depth. The effect is especially perceivable in view 2 and view 3 in fig.2.
On basis of the tone mapping operator, the visible contrast and brightness ratios are reproduced and appear very similar to the images of the real factory. The color temperature of the factory could be approximated using the color grading method mentioned in 5.6. In combination with light scattering of the CLPVs a plausible pigmentation of indirect reflected light is obtained and essential for the atmosphere of the visualization as a whole. In conclusion, a plausible and convincing overall quality of the illumination for virtual environments is achieved in comparison to the images of the real factory. Nevertheless, small discrepancies can be recognized in detail, but due to the diffuse ambient environment inside the factory these artifacts are not disturbing, since the human perception is less sensitive to errors in low frequency illumination environments.

6.2 Performance

An average performance of 6 frames per second is achieved for the test scene in a resolution of 1280x1024 pixels. Thus, the dynamic digital factory can be interactively explored and manipulated. The computational effort of the several stages of the proposed pipeline for the presented views including a detailed listing of the incorporated methods is depicted in fig. 3.

[1] It should be noted that geometrical discrepancies can occur, due to inconsistencies of the model and the real factory

[2] Applicable in combination with the Virtual Texturing technology of [28]

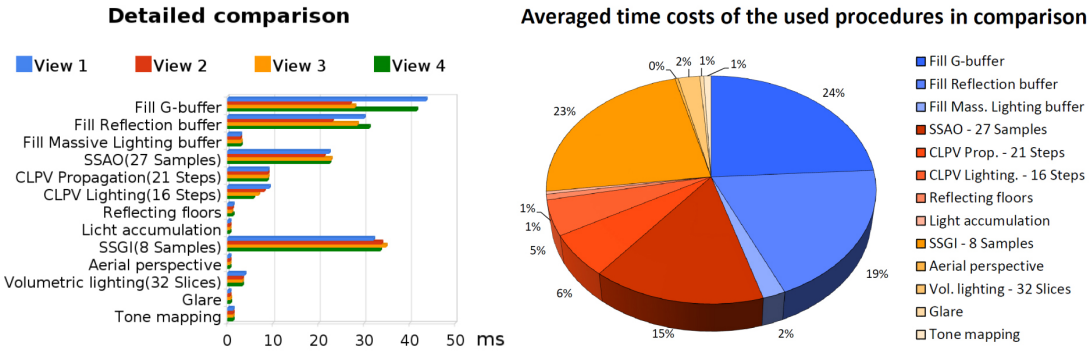


Figure 3: Left: Detailed performance analysis of the several pipeline stages for the considered views. Right: All blue-colored pipeline steps belong to the fill buffer stage, all red-colored to the light accumulation stage and all yellow-colored to the post-processing stage.

Fill Buffer Stage The fill buffer stage with the blue-colored steps in the graph seen on the right side of fig. 3 contains the two time drawing of the scene and therefore is the most expensive stage of the three pipeline stages with an average of 66.06ms. For the filling of the G-Buffer and the Reflection Buffer for the different views strong fluctuations can be noticed in the graph on the left side of fig. 3. These can be explained by the amount of geometry inside the view frustum. In view 2 and view 3 ind fig. 2 a lot of geometry is occluded by the machinery close to the camera, thus the visibility guided renderer can discard a lot of geometry inside the frustum. In average the G-Buffer is filled within 34.97ms, whereas the Reflection Buffer within 28.15ms due to the fact that only a part of the scene covered by the mirror effect has to be rendered. The injection into the CLPVs is independent of the scenes complexity and image resolution and in consequence very performant with merely taking 2.93ms.

Light Accumulation Stage The light accumulation stage with the red-colored steps in the graph is with an average of 40.28ms the second most expensive step, whereas the computation of SSAO with an average of 22.19ms by using 27 samples represents the biggest computational effort. The SSAO performance mainly depends on the image resolution and the amount of used samples. The effort for the light propagation throughout the CLPVs depends on the number of propagation steps and the grid resolution and therefore constantly takes 8.87ms in all views with $21+10+5$ propagations steps and a grid resolution of $48x48x48$. The illumination by the CLPV information depends on the image resolution and the amount of propagation steps in reflection direction through the CLPV to approximate glossy surfaces, which is further denoted as glossy samples. The CLVP performance thus depends on the amount of glossy or reflective areas in the image, which is noticeable on the fluctuations in the timings for the different views, e.g. in view 1 and view 2 more glossy surfaces are visible than in view 3. The average timing is 7.47ms for 16 glossy samples. The last "'light accumulation"' step in this stage, which should not be confused with the whole light accumulation stage, is dependent on the image resolution and includes the multiplication of the accumulated light in the LAB with the material color and the ambient occlusion term within an average of 0.6ms.

Post-Processing Stage The post-processing stage with the yellow-colored steps in the graph takes an average of 39.3ms. SSGI is included in this stage, since it is applied after the multiplication with the material color. The average timing for SSGI is 33.55ms by using eight samples and a kernel size of eight. SSGI depends on the image resolution and the amount of used samples. The computational effort for the color grading is with 0.5ms very small, as well as the effort for the glare with 0.6ms and the tone mapping with 1.2ms. All three methods depend on the image resolution. Participating media lighting effects take an average of 3.43ms by using 32 slices, whereas the effort mainly depends on the number of the processed pixels and the number of slices to approximate the volume.

7 BENEFITS

The presented method can handle dynamic objects and dynamic light sources in massive data sets. Almost all incorporated techniques are independent of the scene complexity, except the reflections on the ground plate, allowing the interactive exploration of massive virtual worlds with an enhanced visual quality and pave the way for high quality desktop VR. Massive light sources can be handled and no preprocessings[3] are needed for all incorporated methods, keeping a short time to production and market. Our scenarios need no manual preparation times since massive amounts of data can be handled by a visibility guided renderer and light sources

[3] except the data export from the CAD tool to the VGR data base

are automatically set during the data export of the CAD tools by looking up the light source geometry with an identifier.

8 CONCLUSION

In this paper we presented an efficient massive lighting pipeline for the interactive computation of a plausible illumination for massive data sets of the dynamic digital factory illuminated by massive amounts of light sources. No preprocessing steps are necessary, keeping a short time to production and market. The key points for a plausible illumination computation of a factory are identified by a detailed analysis and exemplary shown on basis of images of a real factory. The results of our proposed method are evaluated against the key points and the real images and can confirm the plausibility of the computed illumination. Finally, a detailed performance evaluation of the whole pipeline is presented, including the separate listing of all incorporated techniques to identify the main bottlenecks and connecting factors for future research.

ACKNOWLEDGEMENTS

This work was partially funded by the AVILUS project of the german 'Bildungsministerium für Bildung und Forschung'(BMBF).

REFERENCES

[1] Stephens A., Boulos S., Bigler J., Wald I., and Parker S.G. An Application of Scalable Massive Model Interaction using Shared Memory Systems. In *Proceedings of the 2006 Eurographics Symposium on Parallel Graphics and Visualization*, 2006.

[2] M. Deering, S. Winner, B. Schediwy, C. Duffy, and Ne Hunt. The triangle processor and normal vector shader: a vlsi system for high performance graphics. In *SIGGRAPH '88: Proceedings of the 15th annual conference on Computer graphics and interactive techniques*, 1988.

[3] David Drascic and Paul Milgram. Perceptual issues in augmented reality. In *SPIE Volume 2653: Stereoscopic Displays and Virtual Reality Systems III*, pages 123–134, 1996.

[4] Wolfgang Engel. Wolfgang engels blog, article: Light prepass, http://diaryofagraphicsprogrammer.blogspot.com/2008/03/light-pre-pass-renderer.html, 2010.

[5] Dominic Filion and Rob McNaughton. Effects & techniques. In *SIGGRAPH '08: ACM SIGGRAPH 2008 classes*, pages 133–164, New York, NY, USA, 2008. ACM.

[6] Wald I., Dietrich A., Benthin C., Efremov A., Dahmen T., Günther J., Havran V., Slusallek P., and Seidel H.-P. A Ray Tracing based Virtual Reality Framework for Industrial Design. In *Proceedings of the 2006 IEEE Symposium on Interactive Ray Tracing*, 2006.

[7] J. Isidoro. Shadow mapping: Gpu-based tips and techniques. *Game Developer Conference (GDC) 2006 presentation*, 2006.

[8] Henrik Wann Jensen. *Realistic image synthesis using photon mapping*. A. K. Peters, Ltd., Natick, MA, USA, 2001.

[9] James T. Kajiya. The rendering equation. *SIGGRAPH Comput. Graph.*, 20(4):143–150, 1986.

[10] Anton Kaplanyan. Light propagation volumes in cryengine 3. In *SIGGRAPH '09: ACM SIGGRAPH 2009 Courses*, New York, NY, USA, 2009. ACM.

[11] Anton Kaplanyan and Carsten Dachsbacher. Cascaded light propagation volumes for real-time indirect illumination. In *I3D '10: Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games*, 2010.

[12] D. Kasik, B. Brüderlin, M. Heyer, and S. Pfützner. Visibility-guided rendering to accelerate 3d graphics hardware performance. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, 2007.

[13] M. Kawase. Frame buffer postprocessing effects in double-s.t.e.a.l (wreckless). *Game Developer Conference (GDC) 2003 lecture, San Jose, CA.*, 2003.

[14] Hayden Landis. Production-ready global illumination. *In SIGGRAPH 2002 Course Note #16: RenderMan in Production*, pages 87–102, 2002.

[15] M. MITTRING. A bit more deferred-cryengine3. *Triangle Game Conference'09*, 2009.

[16] Martin Mittring. Finding next gen: Cryengine 2. In *SIGGRAPH '07: ACM SIGGRAPH 2007 courses*, pages 97–121, New York, NY, USA, 2007. ACM.

[17] John Murray. Some perspectives on visual depth perception. *SIGGRAPH Comput. Graph.*, 28(2):155–157, 1994.

[18] Steven G. Parker, James Bigler, Andreas Dietrich, Heiko Friedrich, Jared Hoberock, David Luebke, David McAllister, Morgan McGuire, Keith Morley, Austin Robison, and Martin Stich. Optix: A general purpose ray tracing engine. *ACM Transactions on Graphics*, August 2010.

[19] Franz Peschel, Fabian Scheer, and Stefan Müller. Interactive plausible illumination for the digital factory. In *Joint Virtual Reality Conference of EuroVR - EGVE - VEC*, 2010.

[20] E. Reinhard, M. Stark, P. Shirley, and J. Ferwerda. Photographic tone reproduction for digital images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. ACM, 2002.

[21] T. Ritschel, T. Grosch, M. H. Kim, H.-P. Seidel, C. Dachsbacher, and J. Kautz. Imperfect shadow maps for efficient computation of indirect illumination. *ACM Trans. Graph.*, 27(5):1–8, 2008.

[22] T. Ritschel, T. Grosch, and H.-P. Seidel. Approximating dynamic global illumination in image space. In *I3D '09: Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pages 75–82, New York, NY, USA, 2009. ACM.

[23] Fabian Scheer and Michael Keutel. Screen space ambient occlusion for virtual and mixed reality factory planning. *Journal of WSCG*, 18(1-3), 2010.

[24] J. Selan. Using lookup tables to accelerate color transformations. In *GPU Gems 3, Chapter 24*, 2007.

[25] Oles Shishkovtsov. Deferred shading in s.t.a.l.k.e.r. In Matt Pharr and Randima Fernando, editors, *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pages 143–166. Addison-Wesley Professional, 2005.

[26] M. Valient. Deferred rendering in killzone 2, develop conference 2009, http://www.guerrilla-games.com/publications/dr_kz2_rsx_dev07.pdf, 2010.

[27] Michal Valient. Stable rendering of cascaded shadow map. In Wolfgang Engel, editor, *Shader X6, Advanced Rendering Techniques*. Course Technology, CENGAGE Learning, 2008.

[28] J.M.P. van Waveren. id tech 5 challenges, from texture virtualization to massive parallelization. In *SIGGRAPH '09: ACM SIGGRAPH 2009 Courses*, 2009.

[29] Rui Wang, Rui Wang, Kun Zhou, Minghao Pan, and Hujun Bao. An efficient gpu-based approach for interactive global illumination. *ACM Trans. Graph.*, 28(3):1–8, 2009.

Example-based Deformation with Support Joints

Kentaro Yamanaka
Waseda University
3-4-1 Okubo
Shinjukuku
Tokyo, Japan, 1698555
kentaro.05-27@suou.waseda.jp

Akane Yano
Waseda University

Shigeo Morishima
Waseda University
shigeo@waseda.jp

ABSTRACT

In character animation field, many deformation techniques have been proposed. Example-based deformation methods are widely used especially for interactive applications. Example-based methods are mainly divided into two types. One is Interpolation. Methods in this type are designed to interpolate examples in a pose space. The advantage is that the deformed meshes can precisely correspond to the example meshes. On the other hand, the disadvantage is that larger number of examples is needed to generate arbitrary plausible interpolated meshes between each example. The other is Example-based Skinning which optimizes particular parameters referencing examples to represent example meshes as accurately as possible. These methods provide plausible deformations with fewer examples. However they cannot perfectly depict example meshes. In this paper, we present an idea that combines techniques belonging to the two types, taking advantages of both types. We propose an example-based skinning method to be combined with Pose Space Deformation (PSD). It optimizes transformation matrices in Skeleton Subspace deformation (SSD) introducing "support joints". Our method itself generates plausible intermediate meshes with a small set of examples as well as other example-based skinning methods. Then we explain the benefit of combining our method with PSD. We show that provided examples are precisely represented and plausible deformations at arbitrary poses are obtained by our integrated method.

Keywords

Example-based, skinning, deformation, pose space deformation, PSD, skeletal-subspace deformation, SSD, support joints

1. INTRODUCTION

Character deformation plays an important role in computer animations. Poor articulated character deformations are easily perceived when we see 3DCG animations. In order to generate desired character deformations, skilled animators spend much time working on tedious processes. Many character deformation methods have been proposed to resolve this problem, but it hasn't been completed yet.

In articulated character deformation, skeleton-based deformation is widely used to model articulated motion because they are intuitive to use. Among skeleton based deformation methods, Skeleton Subspace Deformation (SSD) [MLT88] is the most common technique, which is also called enveloping or smooth skinning, because it is fast to compute and easy to implement. Despite those advantages it brings, large deformations lead to undesirable effects such as very visible loss of volumes near joints. In order to compensate the defects of SSD, example-based methods are widely used especially in interactive applications.

Example-based deformations can be divided into two types – Interpolation and Example-based Skinning. Interpolation type focuses on how to interpolate examples smoothly in a pose space. Example meshes are always precisely represented, but interpolated ones are not always plausible and many examples are needed to generate plausible interpolated meshes at arbitrary poses. In contrast to interpolation, example-based skinning uses examples only to optimize particular deformation parameters such as vertex weights. The advantage is that plausible intermediate meshes at arbitrary poses between examples are obtained, but example meshes themselves are not fully represented.

In this paper, we present an idea that combines these two types and takes advantage of them. First we propose an example-based skinning method called support joint deformation. It optimizes joint transformation matrices introducing virtual joints called "support joints". Then we integrate our method with Pose Space Deformation (PSD) [LCF01],

which is the most widely used in the interpolation methods, and demonstrate how effective our idea is.

The original support joint deformation method was proposed in our former work [YYM09]. We improve the support joint deformation method and combine the method with PSD in this paper in order to precisely represent examples when characters are deformed. To integrate with PSD effectively, we focused on artifacts occurring around a bending joint instead the previous work took all errors between examples into consideration.

This paper is organized as follows: After giving an overview of related work in character animation area in Chapter 2, we explain the theory and the disadvantage of PSD in Chapter 3 and introduce our core idea of example-based skinning with support joint optimization called support joint deformation in Chapter 4. The way how to set vertex weights suitable for support joint deformation is explained in Chapter 5, before we demonstrate the results of our idea and conclude in Chapter 6.

2. RELATED WORK

Skeleton-based deformation

SSD, which is based on a work published by Magnenat-Thalmann et al. [MLT88], has been very widely used for a long time in character animation field and has been adopted by most computer animation software because it is fast to compute, easy to implement and intuitive to use. The deformation of vertex with SSD is simply represented as follows:

$$\mathbf{v}'_i = \left(\sum_{j=1}^{N} w_{ij} \mathbf{T}_j \right) \mathbf{v}_i \qquad (1)$$

where $\mathbf{v}_i$ is the initial position of the i-th vertex of a character mesh and $\mathbf{v}_i'$ is a position after $\mathbf{v}_i$ is deformed. w_{ij} is a vertex weight which means the amount of influence of the j-th joint on the i-th vertex and it is normalized as $\sum_{j=1}^{N} w_{ij} = 1$. $\mathbf{T}_j$ is a transformation matrix of the j-th joint. N is the number of joints in a skeleton.

Though SSD has been very common, there exists an undesirable defect in this method. Artifacts such as "candy-wrapper" should be unavoidable. They are caused by the very visible loss of volumes when a character mesh is largely deformed.

Many works have been presented to overcome these artifacts. Rohmer et al. proposed a geometrically volume-preserving deformation method with interesting shape controls [RHC09]. Kavan et al. proposed spherical blend skinning [KZ05] and dual quaternion blend skinning [KCZO07] [KCZO08] to realize as rigid deformations as possible.

Example-based deformation

Example-based deformation is one of the attempts to get over the artifacts of SSD. It is often used in interactive application and is roughly divided into two types. The first is interpolation and the second is parameter-optimization. In this paper, we call the latter type example-based skinning. PSD [LCF01] is the most famous work in the interpolation group. It interpolates examples by Radial Basis Function (RBF) in a pose space using joint angles of SSD as deformation parameters. After PSD was proposed, many example-based interpolation methods have been published. "Shape by example" by Sloan et al. enabled extrapolation by adding low order linear polynomials to PSD [SRC01]. Kry et al. proposed a method suited to commercial graphics hardware reducing data of interpolation with PCA [KJP02]. Weighted Pose Space Deformation (WPSD) [KM05] [RLN06] generates arbitrary plausible intermediate poses with fewer examples. Some attempts to use various kinds of measured 3D data in example-based deformation have been presented [ACP02] [KM05]. Provided example meshes are always entirely represented. However, a large set of examples is needed to generate plausible meshes at arbitrary poses.

Example-based skinning needs smaller set of examples than interpolation, because given examples are used only to optimize some particular deformation parameters. Weber et al. proposed their original skeleton based deformation method and, in the paper, they additionally optimized vertex weights from examples in order to add "context" to the results [WSLG07]. Multi-weight enveloping (MWE) published by Wang et al. also optimizes vertex weights [WP02]. They introduced a novel vertex weight which had a value for each element of transformation matrix $\mathbf{T}_j$, instead of w_{ij} in equation (1), and described how to optimize the vertex weights. Mohr et al. proposed additional joints [MG03]. First they optimized values of vertex weights from examples and then they placed additional joints where some artifacts still occurred comparing to examples. It can be said that they optimized vertex weights w_{ij} and a number of joints N in equation (1). Shi et al. [SZTDVG08] optimized parameters for simulation using examples and characters were deformed according to the simulation. We propose an example-based skinning method which optimizes transformation matrices $\mathbf{T}_j$ in equation (1) using support joints.

Vertex weights

Vertex weights play an important role in our method as well as in the other skeleton based deformation methods. Baran et al. [BP07] solved heat equilibrium over character surface to determine vertex weights. They adopted heat equilibrium to satisfy three conditions below. First, vertex weights should

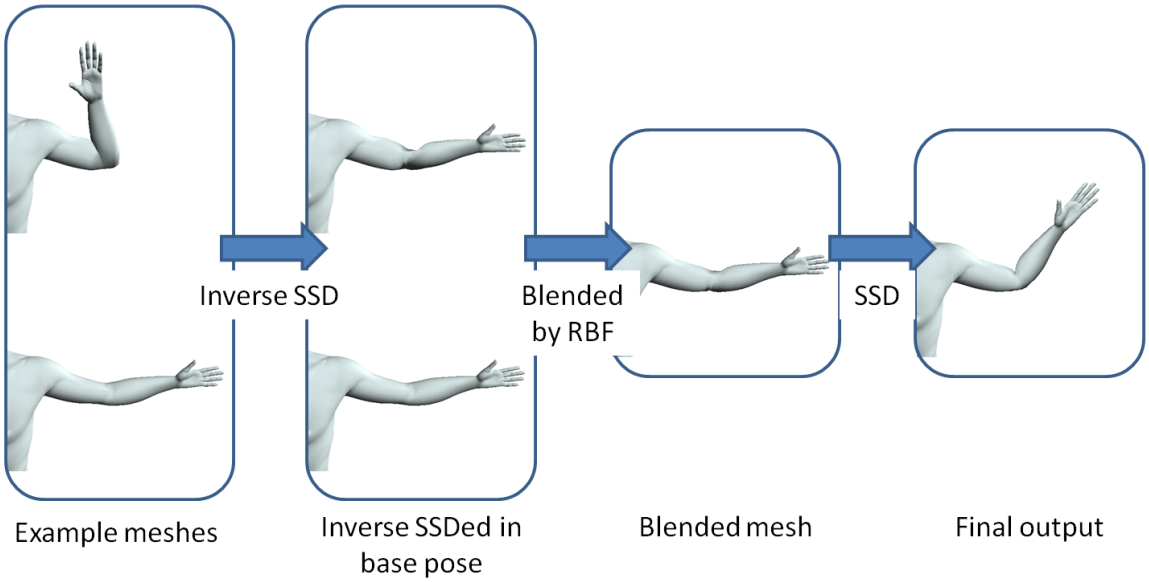


Figure1. The process of PSD

not depend on the mesh resolution. Second, the weights vary smoothly along the surface. Finally, the width of a transition between two bones meeting at a joint should be roughly proportional to the distance from the joint to the surface to avoid folding artifacts. They satisfied those conditions. However, the method needs an initialization of the weight before starting the diffusion smoothing. They suggested to initialize the values to $1/d^2$ when the segment linking the vertex to the bone lies entirely within the mesh volume, and zero otherwise. This automatical discontinuous definition works in most cases, but gives a bad initial value of the skinning weights if the mesh exhibits large non-convexities. Small variations in vertex positions may change dramatically the initial values when the line segment passes close to the mesh boundary. Rohmer et al. compensated this disadvantage and satisfied the conditions above by determining vertex weights according to geodesic volumetric distance [RHC09].

Our previous work [YYM09] defined a distance field μ on a surface mesh based on geodesic distance and determined the vertex weights according to the field. Computing μ is, however, costly because all-pair distances should be calculated and the vertex weights are recalculated at every step of transformation matrices optimization, which also needs much computing time.

3. POSE SPACE DEFORMATION

PSD is a hybrid approach that combines SSD and morphing. Various example meshes are deformed into the "base pose" with inverse SSD, and the resulting meshes are morphed and then deformed with SSD. The process of PSD is shown in Figure1.

Let $\mathbf{v}_{i,p}$ be the position of vertex i of the p-th example. The p-th example mesh is first transformed into its "base pose":

$$\mathbf{v}_{i,p}^{b} = \left(\sum_{j=1}^{N} w_{ij} \mathbf{T}_{j,p} \right)^{-1} \mathbf{v}_{i,p} \tag{2}$$

where $\mathbf{v}_{i,p}^{b}$ is the position of vertex i of the p-th example in its base pose, w_{ij} be the weight value of joint j on a vertex i and $\mathbf{T}_{j,p}$ is the transformation matrix of joint j of the p-th example. Let $s_p(\mathbf{P})$ be the weight value for the interpolation at an arbitrary pose $\mathbf{P}$, satisfied with conditions as follows:

$$\begin{aligned} &\textstyle\sum_{p=1}^{n_{pose}} s_p(\mathbf{P}) = 1, \\ &s_p(\mathbf{P}_k) = 1 \qquad (p = k), \\ &s_p(\mathbf{P}_k) = 0 \qquad (p \neq k) \end{aligned} \tag{3}$$

where $\mathbf{P}_k$ is the k-th example pose and n_{pose} is the number of example poses. $s_p(\mathbf{P})$ is resolved using Radial Basis Function (RBF) as follows:

$$s_p(\mathbf{P}) = \sum_{p=1}^{n_{pose}} c_{p,k} \phi(d(\mathbf{P} - \mathbf{P}_k)) \tag{4}$$

where $c_{p,k}$ is a constant and $\phi(\mathbf{r}) = \exp(-\mathbf{r}^2/2D^2)$ is used in this paper. D is a parameter whose value is determined by users and D = 1.0 in this paper. If D becomes smaller, interpolation weight for the nearest example pose becomes bigger. $d(\mathbf{P} - \mathbf{P}_k)$ is a distance from an arbitrary pose $\mathbf{P}$ to an example pose $\mathbf{P}_k$ in an axis-angle manner. Then, each example surface in the base pose is interpolated by using a morphing method:

$$\mathbf{u}_i^b = \sum_{p=1}^{n_{pose}} s_p(\mathbf{P}) \mathbf{v}_{i,p}^b \tag{5}$$

where $\mathbf{u}_i^b$ is the interpolated vertex position in a base pose. Finally, the morphed surface is deformed with

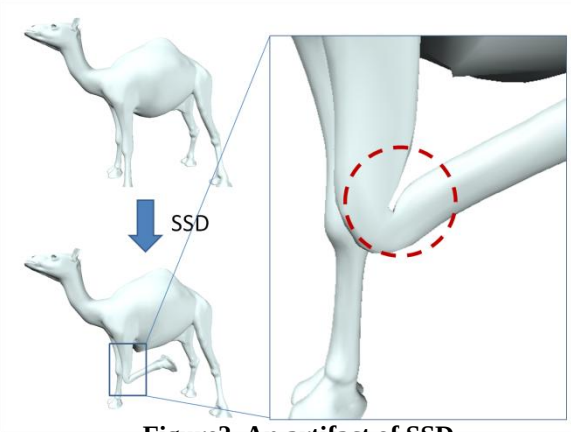


Figure2. An artifact of SSD
Large deformations frequently lead to very visible loss of volume around joints. That causes artifacts of PSD as well.

SSD:

$$\mathbf{u}_i = \left(\sum_{j=1}^{N} w_{ij} \tilde{\mathbf{T}}_j \right) \mathbf{u}_i^b \qquad (6)$$

where $\mathbf{u}_i$ is the vertex position of the resulting deformed surface, $\tilde{\mathbf{T}}_j$ is the transformation matrix that is calculated by interpolating the matrices of examples using $s_p(\mathbf{P})$.

As explained above, PSD is largely based on SSD. With SSD, however, large deformations lead to undesirable results such as very visible loss of volume near joints (See Figure2). That also causes artifacts in PSD. The artifacts of PSD are mainly caused by the steps of the inverse SSD and SSD in the process (See Figure1). Figure3 shows an example of artifacts in PSD due to SSD. When examples are deformed into the base pose with SSD (inverse SSD), artifacts frequently occur. Then the artifacts remain on the blended mesh. Besides, more artifacts may happen at the last SSD step. Figure1 shows mechanism how artifacts of PSD occur as well. Users have been forced much more tedious works preparing additional examples for the results to look better. In order to avoid this otiose task, a deformation method which has fewer artifacts should be adopted instead of SSD to improve the quality of PSD. Then we propose such a method in the next chapter.

4. EXAMPLE-BASED SKINNING WITH SUPPORT JOINTS

Motivation

In order to overcome artifacts of PSD derived from the defects of SSD such as apparent loss of volumes around a joint when the joint is largely bent, we considered that example-based skinning method is the most suitable and efficient alternative for SSD because there are already prepared examples and it usually takes much time for users to prepare additional ones. Therefore this chapter presents example-based skinning method which reduces artifacts around a bent joint.

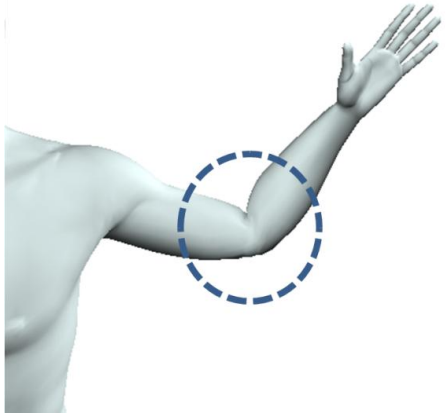


Figure3. An artifact of PSD because of SSD

Example-based skinning optimizes particular deformation parameters from examples for the results to look good. Wang et al. [WP02] introduced a novel vertex weight which has one value for each of twelve elements of transformation matrix in equation (1), and optimized them. Then it is called Multi Weight Enveloping (MWE). After optimizing vertex weights Mohr et al. optimized the number of joints each of which has a special function such as scaling in order to reduce remaining artifacts and represent examples as correct as possible. It hasn't been presented yet a method which optimizes transformation matrices, or improves behavior of joints from examples though it is often said that the poor structure of a traditional hierarchical skeleton is one of the problems that causes various artifacts of skeleton based deformation. A method which optimizes transformation matrices and reduces the loss of volumes around a bent joint with SSD is thought to be a solution to reduce the artifacts of PSD.

Support Joint Deformation

In order to realize optimization of transformation matrices, a set of support joints **S** is defined besides the traditional hierarchical skeleton **H**. **S** doesn't have hierarchical structure and each joint of **S** can move around the corresponding joint of **H** according to the rule explained later, in contrast to that **H** has a hierarchical structure and the distances between joints are constant.

In our proposing method, it is assumed that example meshes and corresponding joint angles of **H** are given. After a base mesh is deformed according to the given joint angles of **H** in an example pose with SSD, transformation matrices of **S** are calculated to minimize error function $\xi(\mathbf{S})$ described below:

$$\xi(\mathrm{S}) = \sum_i \left\| \mathbf{v}_i^T - \left(\sum_{j=1}^{N} w_{ij}^S \mathbf{T}_j^S \right) \mathbf{v}_i^{SSD} \right\| \qquad (7)$$

where i is the index of vertex. $\mathbf{v}_i^T$ is a position of vertex i of a given example pose. w_{ij}^S means a vertex weights of the j-th joint of **S** on vertex i, which is described in Chapter 5 in detail. $\mathbf{T}_j^s$ is the transformation matrix of the j-th joint of **S** that should be optimized, and $\mathbf{v}_i^{SSD}$ is a vertex deformed by SSD according to the given joint angles. Therefore $\xi(\mathbf{S})$ returns a difference between a target mesh and a deformed mesh.

In this paper, there are rules how joints of **S** moves around joints of **H** to obtain the optimal position and the optimal transformation matrix efficiently. Discrete values are set on x-axis linked coordinate of the parent joint of **S** and iterative search algorithm which minimize$\xi(\mathbf{S})$ is solved. Figure4 shows how **S** moves. The rules are described below.

- ✓ The j-th support joint moves along a line from the bent joint j of **H** to the parent joint j-1.
- ✓ First, the j-th support joint is translated along the axis, and then it is rotated to the direction which a virtual bone connects j-th support joint of **S** and j+1-th joint of **H**.

According to the manner above, transformation matrix $\mathbf{T}_j^S$ is calculated.

Using the calculated transformation matrix above, plausible intermediate meshes are obtained at arbitrary poses. According to the pose desired by uses, the intermediate transformation matrix is calculated by interpolating the transformation matrices $\mathbf{T}_j^s$ for example poses. $s_p(\mathbf{P})$ in equation (5) is used to blend transformation matrices $\mathbf{T}_j^s$ of example poses. The result of support joint deformation can be obtained when $\mathbf{v}_i$, w_{ij}, and $\mathbf{T}_j$ in equation (1) are replaced by $\mathbf{v}_i^{SSD}$, w_{ij}^S, and interpolated $\mathbf{T}_j^s$.

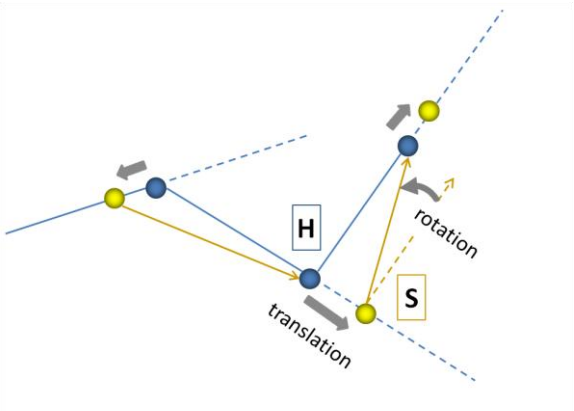


Figure4. The manner how support joints move

Blue: Hierarchical skeleton H, Yellow: A set of support Joints S. The j-th support joint moves on a line which connects j-th and j-1th hierarchical joint, and the direction of the j-th support joint are determined from the position of the j+1th hierarchical joint.

Figure5. The result of support joint deformation compared to SSD

Left: SSD, Right: support joint deformation
Both meshes are deformed with the same joint angles.

Figure5 shows how much our proposing example-based skinning reduces the artifacts. The artifacts of SSD could be sufficiently improved by our proposing support joint deformation.

5. VERTEX WEIGHTS

In skeleton based deformation vertex weights play a very important role and it's also true in our method. Especially for SSD, users often set vertex weights manually using painting tool in commercial animation software. However, it takes much time for users to set vertex weights, while there often exist puzzling cases in which no set of vertex weights can avoid an artifact. Then many techniques to decide vertex weights automatically have been proposed.

Example-based skinning methods decide optimal vertex weights to represent the examples the best referencing a set of the given example meshes and the corresponding skeletons. Mohr et al. optimized vertex weights before they introduced additional joints [MG03]. Weber et al. optionally optimized vertex weights from examples to add "context" to the results after they deformed characters with their original deformation method [WSLG07].

Though support joint deformation described in the previous chapter is also classified into example-based skinning, vertex weights in our method are determined not based on examples but geometrical information. The reason is that we need to determine two types of vertex weights. The first one is for SSD represented as w_{ij} in equation (1). The second one is for support joint deformation represented as w_{ij}^S in equation (7). w_{ij} can be determined from examples like other example-based skinning methods because values of the other variables are already known. On the other hand, w_{ij}^S cannot be optimized by examples because the transformation matrix $\mathbf{T}_j^S$ is unknown and it is required to be calculated from geometrical information of examples. Because of the efficiency, we adopt a technique which is able to calculate both different vertex weights from geometrical informa-

tion. Both vertex weights are required to meet conditions below.

- Vertex weights vary smoothly and continuously along the surface.
- Vertex weights avoid artifacts caused by incorrect association between a joint and vertices.
- About w_{ij}, a transition of vertex weights around joints between two bones is roughly proportional to the distance from the joint to the surface.
- About w_{ij}^S, values of weights are high (close to 1) around the bent joint and converge to 0 smoothly according to the distance from the joint.

The simplest technique to determine vertex weights referencing only geometrical information of a mesh is to calculate based on Euclidean distance from a joint to vertices. It can satisfy the first condition although it cannot meet the second and often causes artifacts that attach weights from wrong joint to a vertex because topology of character mesh is unconsidered. In this paper we adopt the technique which Baran et al. poposed [BP07] to determine both two types of vertex weights.

Calculation of geodesic volumetric distance

Character mesh is required to be filled with voxels (Figure6) prior to the measurement of geodesic volumetric distance. We assume that a voxelized model is already pre-computed. In order to calculate the distance from a joint or a bone to vertices, a voxel which includes the joint (or bone) inside and a voxel which includes a vertex inside need to be specified. The geodesic volumetric distance from the joint (or bone) voxel to vertex voxels are calculated using dijkstra's algorithm applied to three dimensions.

STEP1: Initialize $g(v) = \infty$ about all vertex voxels.

STEP2: Select a base voxel b, set $g(b) = 0$, and insert b to VLIST.

STEP3: Take the vertex voxel v which has the smallest g(v) in VLIST and remove it from VLIST.

STEP4: For each vertex voxel v_a adjacent to v, if $g(v) + \text{length}(v, v_a) < g(v_a)$, update $g(v_a) = g(v) + \text{length}(v, v_a)$ and insert (or reinsert) v_a to VLIST, where length(v, v_a) is defined as the Euclidean distance from the center of v to the center of v_a. Notice that the adjacency of vertex voxels is determined whether a voxel is a member of a cube, which consists of 27 (= 3*3*3) voxels whose center is v, or not.

STEP5: Repeat Step3 and 4 until VLIST become empty.

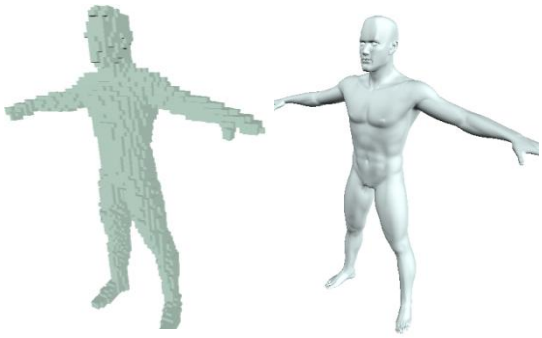


Figure6. A voxelized character mesh
Left: A voxelized character mesh, Right: The corresponding character mesh

Figure7. Geodesic volumetric distance from a forearm bone
Red means that the distance from a voxel including the bone to the vertex is small. Blue means the vertex is distant from the bone voxel.

Geodesic volumetric distance calculated with this algorithm is shown in Figure7.

Determination of vertex weights

In this paper, two types of vertex weights are determined using the calculated distance above. The first weights are designed for SSD. It is required to meet the first, the second and the third condition previously mentioned. To satisfy the third condition, the distances from bones are used to determine the vertex weights. Therefore, bone voxels are set as base voxels in the calculation process. Figure8 shows vertex weights from forearm bones. A vertex weight is determined by reciprocal ratio of squared distances from the nearest and the second nearest bone, and the result is normalized.

The second weights are for support joint deformation. It is required to meet the first, the second and the fourth condition and therefore distance from the bent joint is adopted. The vertex weights for support joint deformation are determined by multiplication of a gaussian function of a distance from a joint and the error of a vertex position on a base mesh deformed by SSD from a vertex position on an example mesh. Note that, when multiplied, the errors are normalized to let the maximum value be 1. Figure9 shows the resulting vertex weights for support joint deformation. Vertex weights for support joint deformation are

determined referencing the distance from the joint and the error between a SSDed mesh and the corresponding example mesh.

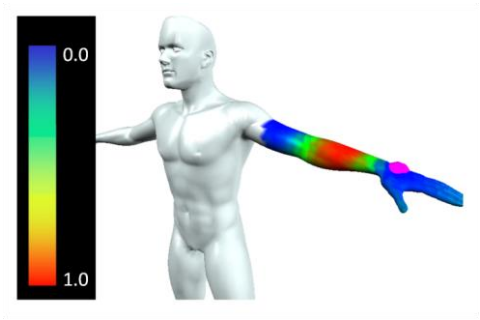


Figure8. Vertex Weights of the forearm bone for SSD
Red: a vertex where $\mathbf{w_{ij}} = \mathbf{1.0}$, Blue: the value of $\mathbf{w_{ij}}$ is close to 0.0, White: precisely $\mathbf{w_{ij}} = \mathbf{0.0}$

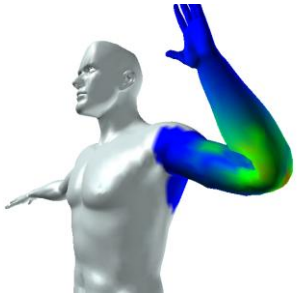


Figure9. Vertex Weights of the elbow joint for support joint deformation
Red: a vertex where $\mathbf{w_{ij}} = \mathbf{1.0}$, Blue: the value of $\mathbf{w_{ij}}$ is close to 0.0, White: precisely $\mathbf{w_{ij}} = \mathbf{0.0}$

6. RESULTS AND CONCLUSIONS

Integrated method

Finally we present a method which combines PSD as an interpolation and support joint deformation as an example-based skinning. It is simply explained as following. The inverse SSD step in the process of PSD is replaced by the inverse support joint deformation, and the last SSD step is replaced by support joint deformation. First, transformation matrices $\mathbf{T_j^S}$ of bent joints of examples are optimized. Second, examples are deformed into base pose by inverse support joint deformation. Note that inverse support joint deformation consists of two steps. First step is inverse SSD and the second step is multiple inverse of $\mathbf{T_j^S}$ to a deformed mesh. Then deformed examples in base pose are morphed. Lastly, The morphed mesh is deformed with support joint deformation. Though our method needs additional computational time to PSD in order to optimize the transformation matrices as pre-processing, the deformation is executed as fast as PSD because we only changed SSD to support joint deformation.

Result

Figure10 shows our method compared with PSD. It demonstrates that our integrated technique of PSD and support joint deformation entirely represents the example poses as PSD does and generates arbitrary plausible intermediate poses between each example. It means that plausible results at arbitrary intermediate poses can be obtained with fewer examples than PSD because PSD needs more examples to improve the results. Therefore, it can be said that our proposing method successfully takes advantages of interpo-

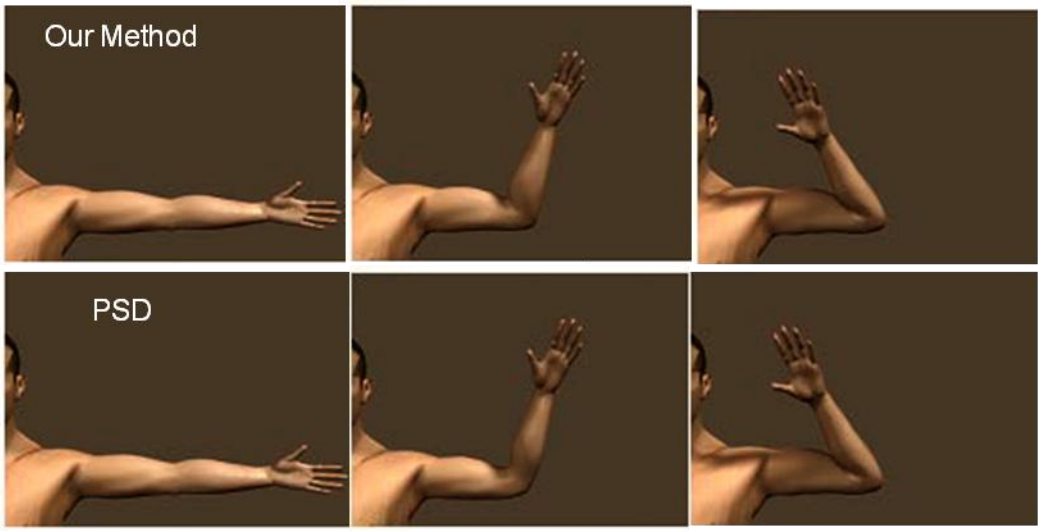


Figure10. The result of our method compared with PSD
Upper: Our proposing method that combines PSD and our original Support Joint Deformation.
Lower: PSD
Left and right images show that our method precisely represents the example poses as PSD does, and the middle images demonstrate our method is able to generate more plausible intermediate poses than PSD.

lation and example-based skinning.

Future work

We would like to integrate support joint deformation into Weighted Pose Space Deformation (WPSD) [KM05] [RLN06] for the results to look better with smaller set of examples than combining with PSD. This paper regulated motion of support joints as described in Chapter4. We would like to deregulate the manner and reduce all kinds of artifacts. Geodesic volumetric distance is adopted to measure distance between a joint (bone) and a vertex when vertex weights are determined. Instead, interior distance [RLF09] can be also employed to reduce the computing time of pre-processing.

7. REFERENCES

[ACP02] ALLEN, B., CURLESS, B., and POPOVIC, Z. Articulated body deformation from range scan data. Proceedings of the 29th annual conference on Computer graphics and interactive techniques, pp.612-619, 2002.

[BP07] BARAN, I., and POPOVIC, J. Automatic rigging and animation of 3d characters. ACM SIGGRAPH 2007 papers, p.72, 2007.

[KCZO07] KAVAN, L., COLLINS, S., ZARA, J., and O'SULLIVAN, C. Skinning with dual quaternions, Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games. ACM Press, pp.39-46, 2007.

[KCZO08] KAVAN, L., COLLINS, S., ZARA, J., and O'SULLIVAN, C. Geometric skinning with approximate dual quaternion blending. ACM Transaction on Graphics, 27, 4, 2008.

[KJP02] KRY, P., JAMES, D. L., PAI, D. K.. EigenSkin: Real time large deformation character skinning in hardware. Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.153-159. 2002.

[KM05] KURIHARA, T., and MIYATA, N. Modeling deformable human hands from medical images. Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation (Aire-la-Ville, Switzerland), Eurographics Association, pp.355-363, 2005.

[KZ05] KAVAN, L., and ZARA, J. Spherical blend skinning: a real-time deformation of articulated models. Symposium on Interactive 3D Graphics and Games, pp.9-16, 2005

[LCF01] LEWIS, J. P., CORDNER, M., and FONG, N. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. Proceedings of the 27th annual conference on Computer graphics and interactive techniques, pp.165-172, 2000.

[MG03] MOHR, A., and GLEICHER, M. Building efficient, accurate character skins from examples. ACM SIGGRAPH 2003 Papers, pp.562-568, 2003.

[MLT88] MAGNENAT-THALMANN, N., LAPERRIERE, R., and THALMANN, D. Joint-dependant local deformations for hand animation and object grasping. Proceedings on Graphics interface '88, pp.26-33, 1988.

[RLF09] RUSTAMOV, R., LIPMAN, Y., FUNKHOUSER, T. Interior Distance Using Barycentric Coordinates. Computer Graphics Forum (Symposium on Geometry Processing) 28(5), July, 2009

[RLN06] RHEE, T., LEWIS, J., and NEUMANN, U. Real-time weighted pose-space deformation on the GPU. Computer Graphics Forum 25, 3. 2006

[RHC09] ROHMER, D., HAHMANN, S., and CANI, M.-P. Exact volume preserving skinning with shape controls. Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, pp.83-92, 2009.

[SRC01] SLOAN, P.-P., ROSE, C., and COHEN, M. Shape by example. Proceedings of the 2001 symposium on Interactive 3D graphics, pp.135-143, 2001.

[SZTDVG08] SHI, X., ZHOU, K., TONG, Y., DESBRUN, M., BAO, H., and GUO, B. Example-based Dynamic Skinning in Real Time. ACM Trans. on Graphics, August 2008.

[YYM09] YAMANAKA, K., YANO, A., MORISHIMA, S. Example Based Skinning with Progressively Optimized Support Joints. ACM SIGGRAPH Asia 2009 poster, December 2009.

[WP02] WANG, X. C., and PHILLIPS, C. Multi-weight enveloping: Least-squares approximation techniques for skin animation. Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.129-138, 2002.

[WSLG07] WEBER, O., SORKINE, O., LIPMAN, Y., and GOTSMAN, C. Context-aware skeletal shape deformation. Comp. Graph. Forum 26, 2007.

Integration of Reconstruction Error Obtained by Local and Global Kernel PCA with Different Role

Kazuhiro Hotta
Meijo University
1-501 Shiogamaguchi, Tenpaku-ku, Nagoya 468-8502, JAPAN
Email: kazuhotta@meijo-u.ac.jp

ABSTRACT

This paper presents a scene classification method using the integration of the reconstruction errors by local Kernel Principal Component Analysis (KPCA) and global KPCA. There are some methods for integrating local and global features. However, it is important to give obvious different role to each feature. In the proposed method, global feature with topological information represents the rough composition of scenes and local feature without position information represents fine part of scenes. Experimental results show that accuracy is improved by using the reconstruction errors obtained from the different point of views. The proposed method is much better than only local KPCA, global KPCA and linear Support Vector Machine (SVM) of bag-of-visual words with the same basic feature. Our method is also comparable to conventional methods using the same database.

Keywords:
Integration, Local, Global, Kernel PCA, Scene classification

1 INTRODUCTION

In recent years, many local feature based methods has been proposed [1, 2, 3, 4]. Local features are more robust to pose variations [4, 5] and partial occlusion [6] than global features. Although local features have these advantages, local features based methods tend to mis-classify samples which are classified easily by global features. Global features are adequate to extract the rough information and relation with various regions though they are not robust to pose variations and partial occlusion. Thus, if we integrate global features and local features well, the accuracy will be improved.

There are some methods for combining local features with global features. For example, a face detector using SVM with a summation kernel of local and global features was proposed [7]. However, when local and global features are integrated in the level of a kernel function and a detector is constructed by SVM with the kernel, the properties of local and global features may deny each other. Li [5] used holistic image as well as local patches in pose independent face recognition. Although accuracy was improved by using the sum of probabilities by both features, there is a possibility that the both properties are not used sufficiently in the simple summation.

To integrate effectively local and global features, it is important to give each feature to obvious different role (property). There are some methods which gave each feature to different role. Rao [8] proposed a brain model in which global prediction and local complementation were integrated. They reported that end-stopping cell was obtained by this formulation. Murphy [9] integrated local and global features in Bayes theorem to localize objects in images. In this method, global feature was used as context and local features were used as part classifiers. By giving the obvious different role to each feature, localization accuracy was improved.

In recent years, global features were used as contextual information for object detection [9, 10]. However, in these methods, scene category information was not used. If the system recognizes the category of scenes not only global feature of an image, the system can predict the object candidates which are probably included in the scene category. Thus, researchers pay attention to scene category classification problem in recent years [11, 12, 2, 13, 14]. To classify scene category, the rough composition of images is important. In this paper, KPCA of global features represents the composition of images. It is effective for scene classification. However, global feature of an image is easily influenced by the position changes of objects in scenes. In general, the positions of objects in scenes are not static.

Therefore, the sift-invariant similarities by local features should be integrated with the rough composition. To do so, we integrate KPCA of local features without position information and global KPCA. We show that accuracy is improved by integrating the reconstruction errors obtained by both KPCAs with different role.

The proposed method is evaluated using 13 scene category database [15] because many methods were evaluated using this database [11, 12, 2, 13, 14]. We evaluated our method using the same experimental setting with conventional methods. The proposed method achieves more than 82.5% by integrating the reconstruction errors obtained by both KPCAs though only global KPCA and local KPCA achieve below 77%. The accuracy is much better than the linear SVM of bag-of-visual words with the same basic feature. Our approach is also comparable with the conventional methods.

In section 2, the details of the proposed method are explained. Evaluation results using 13 scene database are shown in section 3. Finally, conclusion and future works are described in section 4.

2 PROPOSED METHOD

The proposed method consists of 3 steps. The first step extracts the features from images. In this paper, grid sampling with 16×16 grids is used, and orientation histograms of Gabor features are developed at each grid.

The second step is the local and global KPCAs. In local KPCA, 4 orientation histograms without position information on 2×2 grid are used as a local feature. In global KPCA, orientation histograms with topological information on an image are used. Local KPCA represents the fine part of scenes and global KPCA represents the rough composition of scenes. Note that global KPCA is position dependent and local KPCA is position independent. The third step is the classification by integrating the reconstruction errors in both KPCAs.

In section 2.1, orientation histogram of Gabor features is explained. Local and global KPCAs are explained in section 2.2. Section 2.3 explains the classification by integration of both KPCAs.

2.1 Features for scene classification

In recent years, the effectiveness of orientation histogram [1] for object recognition is reported. We develop the orientation histogram from multi-scale Gabor features because Gabor features are better representation than simple gradient features [13].

First, we define Gabor filters. They are defined as

$$\psi_k(z) = \frac{k_\nu^2}{\sigma^2} \exp\left(\frac{-k_\nu^2 z^T z}{2\sigma^2}\right) \cdot \left(\exp\left(ik^T z\right) - \exp\left(-\sigma^2/2\right)\right), \quad (1)$$

where $z = (y,x)^T$, $k = k_\nu \exp(i\phi) = (k_\nu \cos(\phi), k_\nu \sin(\phi))^T$, $k_\nu = k_{max}/f^\nu$, $\phi = \mu \cdot \pi/8$, $f = \sqrt{2}$ and $\sigma = \pi$. In the experiments, Gabor filters of 8 different orientations ($\mu = \{0,\ldots,7\}$) with 3 frequency levels ($\nu = \{0,1,2\}$) are used. In the following experiments, the norm of real and imaginary parts at each point is used as the output of a Gabor filter. The size of Gabor filters of 3 different frequency levels is set to 9×9, 13×13 and 17×17 pixels respectively.

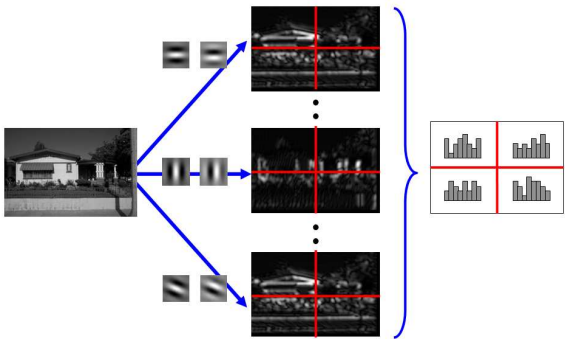


Figure 1: Orientation histogram from Gabor features

Next, we explain how to develop the orientation histogram from the output of Gabor filters. In this paper, the orientation histogram is developed from evenly sampled $M \times M$ grid. Figure 1 shows the example of 2×2 grid with only one scale parameter [1]. First, Gabor features (real and imaginary parts) of 8 orientations are extracted from the input image. The norm of real and imaginary parts at each pixel is computed. Then the orientation histogram with 8 bins at each grid is developed by voting the output value of the maximum orientation at each pixel to the orientation bin. This process is repeated at each scale parameter independently.

In the experiments, we use evenly sampled 16×16 grid, and the orientation histogram of 24 dimensions (= 3 scales $\times$ 8 orientation bins) is developed at each grid. Only 24 dimensional orientation histogram at each grid is too small to classify scenes by using only local features. Thus, we use 4 orientation histograms on 2×2 grid without overlap are used as one local feature. Namely, 64 (= 8×8) local features are obtained from an image. The dimension of a local feature is 96 (= $24 \times 2 \times 2$).

In local KPCA, local features without position information are used. In global KPCA, all 64 local features with position information are used.

2.2 Local and global KPCA with different role

In this section, at first, we explain KPCA and kernel function. After that, local KPCA and global KPCA are explained.

Kernel PCA This section explains KPCA [16, 17] briefly. When data $\{x_1 \ldots, x_L\}$ is given, x is mapped

[1] In the experiments, Gabor features of 3 scale parameters are used.

into high dimensional space by non-linear mapping $\phi(x)$. By applying linear PCA in high dimensional space, non-linear principal components are obtained. Covariance matrix in high dimensional space is computed by

$$C = \frac{1}{L}\sum_{i=1}^{L}\phi(x_i)\phi(x_i)^T. \quad (2)$$

Eigen value problem for KPCA is defined by $\lambda V = CV$ where λ is eigen value and V are eigen vectors. Eigen vectors lie in the span of $\phi(x_1),\ldots,\phi(x_L)$. Therefore, the eigen vector is represented by

$$v = \sum_{i=1}^{L}\alpha_i\phi(x_i), \quad (3)$$

where α_i is the coefficient.

The equation does not change when $\phi(x_k)$ is multiplied to both sides. Then the eigen value problem is changed as

$$\lambda\phi(x_k)^T V = \phi(x_k)^T CV \qquad \text{for all } k = 1,\ldots,L. \quad (4)$$

By substituting eigen vectors shown in equation (3) into equation (4) and using the kernel matrix K where $K_{ij} = \phi(x_i)^T\phi(x_j)$, we obtain the following eigen value problem

$$L\lambda\alpha = K\alpha. \quad (5)$$

By solving the eigen value problem, α is obtained. We have to normalize the obtained α^p for satisfying $v_p^T v_p = 1$ for all $p = 1,\ldots,L$.

An input sample x is mapped into the p-th principal component axis by

$$v_p^T\phi(x) = \sum_{i=1}^{L}\alpha_i^p\, K(x_i,x). \quad (6)$$

The new feature vector in KPCA space is obtained by the weighted sum of similarities with training samples because kernel function computes the similarity with training samples.

Next, moving on to consider the types of kernel function, it is reported that a normalized polynomial kernel gives comparable performance with a Gaussian kernel using optimal parameters [18]. In addition, the parameter dependency of a normalized polynomial kernel is much lower than that of a Gaussian kernel. Since a normalized kernel satisfies Mercer's theorem [19], it is used as the kernel function. The normalized polynomial kernel is defined as

$$\begin{aligned} K(x,y) &= \frac{\phi(x)^T\phi(y)}{||\phi(x)||\;||\phi(y)||}, \\ &= \frac{(1+x^T y)^d}{\sqrt{(1+x^T x)^d\;(1+y^T y)^d}}. \end{aligned} \quad (7)$$

By normalizing the output of a standard polynomial kernel, the kernel value is between -1 and 1. In this paper, all orientation histograms are positive values as explained in section 2.1. Thus, the kernel value takes between 0 and 1 as with a Gaussian kernel. In local KPCA, $d = 5$ is used empirically.

Local KPCA Since the distribution of local features without position information is non-linear, KPCA is appropriate for representing it [4, 20]. In this paper, KPCA is applied to the set of 4 orientation histograms without position information on 2×2 grid, and we call this "local KPCA". Since the norm normalization of an input feature vector improves accuracy [6], the norm of each orientation histogram is normalized before applying local KPCA.

Reconstruction error of $\phi(x)$ by local KPCA can be computed as

$$\begin{aligned} ||\phi(x) - VV^T\phi(x)||^2 &= \phi(x)^T\phi(x) - \phi(x)VV^T\phi(x) \\ &= K(x,x) - ||V^T\phi(x)||^2. \end{aligned} \quad (8)$$

Since we use a normalized polynomial kernel, $K(x,x) = 1$ and the reconstruction error takes the value between 0 and 1. $||V^T\phi(x)||^2$ is called as CLAss-Featuring Information Compression (CLAFIC) [21, 22, 23, 24]. The reconstruction error is also called as Distance From Feature Space (DFFS) [25]. The reconstruction error $K(x_i,x_i) - ||V^T\phi(x_i)||^2$ of i-th local feature x_i is denoted as ε_{li}.

In the classification by using only local KPCA, we compute the sum of reconstruction errors of all local features in an image, and the image is classified to the category which has minimum reconstruction error.

Global KPCA with local summation kernel In this paper, we want to compute the reconstruction error of the i-th local feature x_i from local and global viewpoints, and both reconstruction errors are integrated to improve accuracy. When global KPCA without any devices is applied to the set of all local features of an image, we obtain only the total reconstruction error ε_g and can not obtain the reconstruction error ε_{gi} of the i-th local feature x_i. Therefore, we use the local summation kernel [26] and the expansion of it [20] to compute the reconstruction error of i-th local feature by global KPCA.

Local summation kernel in which local kernels are summarized is defined as

$$\begin{aligned} K_{sum}(x,y) &= \sum_{i}^{N}\phi(x_i)^T\phi(y_i) = \sum_{i}^{N}K(x_i,y_j) \\ &= \phi_g(x)^T\phi_g(y) \end{aligned} \quad (9)$$

where $\phi_g(x) = (\phi(x_1)^T,\ldots,\phi(x_N)^T)^T$ and $x = (x_1^T,\ldots,x_N^T)^T$. Namely, in a local summation kernel, each local feature x_i is mapped into $\phi(x_i)$ and global feature $\phi_g(x)$ is constructed by connecting all $\phi(x_i)$. After that linear PCA

is applied to the set of $\phi_g(x)$ extracted from training images.

If we use a normalized polynomial kernel with 2nd degree as a local kernel, we can compute eigen vectors of primal form directly not dual form. Therefore, we can compute the reconstruction error of i-th local feature by using the eigen vectors of the primal form. Note that dual form is the description using kernel function and primal form uses $\phi(x)$ directly not kernel function.

In normalized polynomial kernel with 2nd degree ($d = 2$ in equation (7)), the dimension of a mapped feature $\phi(x)$ becomes $(nd+2)(nd+1)/2$ when the dimension of an input feature x is nd. For example, the 2 dimensional feature $x = (x_1, x_2)^T$ is mapped into 6 dimensional feature $\phi(x) = (x_1^2/a, x_2^2/a, \sqrt{2}x_1/a, \sqrt{2}x_2/a, \sqrt{2}x_1x_2/a, 1/a)^T$ where a is the norm of the vector $(x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1)^T$. Note that the norm of mapped feature is normalized to 1 in a normalized polynomial kernel. In this paper, the dimension of $\phi(x_i)$ is 4753 because the dimension of a local feature x_i is 96.

The eigen vectors W with the primal form which are obtained by global KPCA with a local summation kernel can be described as

$$W = (w_1, \ldots, w_M), \tag{10}$$

where M is the number of dimension (eigen vectors used) of KPCA space. The p-th eigen vector w_p can be described as

$$w_p = (w_{p1}^T, \ldots, w_{pN}^T)^T. \tag{11}$$

This equation means that each eigen vector is connected the coefficient vectors for $\phi(x_i)$ which is the feature after non-linear mapping of i-th local feature. The dimension of w_{pi} corresponds to $\phi(x_i)$. Thus, a global feature x extracted from an image is mapped into the p-th principal component axis as

$$w_p^T \phi_g(x) \quad = \quad \sum_i^N w_{pi}^T \phi(x_i). \tag{12}$$

Since we use a local summation kernel, inner product between eigen vector and $\phi_g(x)$ can be decomposed into the summation of local inner products.

The difference from local KPCA is the eigen vectors which are determined by using entire feature extracted from an image. Namely, the eigen vectors of global KPCA are position dependent though the eigen vectors of local KPCA are not. Since global KPCA with a local summation kernel is the linear PCA of $\phi_g(x)$, eigen vectors also have relative information with other local regions.

The computation of reconstruction error by global KPCA is easy because $\phi_g(x)$ and $\widehat{\phi_g(x)} = WW^T\phi_g(x)$ can be computed directly by primal form. Since the total reconstruction error $||\phi_g(x_i) - \widehat{\phi_g(x_i)}||^2$ is the sum of reconstruction error of all local features as $\sum_i^N ||\phi(x_i) - \widehat{\phi(x_i)}||^2$, the reconstruction error of the i-th local feature can be computed easily. The reconstruction error of i-th local feature is described as ε_{gi}.

In the classification by using only global KPCA, the total reconstruction error $\sum_i^N \varepsilon_{gi}$ of an image is computed, and the image is classified to the category which has minimum error.

2.3 Classification by inter-complementation

We integrate the reconstruction errors obtained by local and global KPCAs with different role. Figure 2 shows the reconstruction by local KPCA. The i-th local feature x_i (square region in the Figure) is mapped to $\phi(x_i)$ which is shown as the circle in the Figure. The circle on the right side shows the reconstructed feature $VV^T\phi(x_i)$ by local KPCA. The difference between 2 circles is the reconstruction error of i-th local feature.

Figure 3 shows the reconstruction by global KPCA with a local summation kernel. The i-th local feature x_i is mapped to $\phi(x_i)$, and global feature is constructed as $\phi_g(x) = (\phi(x_1)^T, \ldots, \phi(x_N)^T)^T$. The circle on the left side in the Figure shows the global feature $\phi_g(x)$ and the circle on the right side shows the reconstructed feature $WW^T\phi_g(x)$ by global KPCA. As shown in previous section, the total reconstruction error by global KPCA is divided into the reconstruction error at each local feature.

The difference between the reconstruction error by local and global KPCAs is whether position dependent or not. In addition, global KPCA with a local summation kernel uses the relation with various regions though relative information with other regions is not used in local KPCA. Therefore, the integration of both reconstruction errors obtained from the different points of view will improve the accuracy.

To integrate the both reconstruction errors, we use the weighted integration as

$$E = \gamma \sum_i^N \varepsilon_{li} + (1-\gamma) \sum_i^N \varepsilon_{gi}, \tag{13}$$

where γ is the weight. A test image is classified to the category which gives the minimum integration error. If we set γ to 0, the method corresponds to the use of only global KPCA. $\gamma = 1$ means that only local KPCA is used. Experiments demonstrate the effectiveness of our integration method.

3 EXPERIMENTS

In this section, the proposed method is evaluated using the 13 scene database [15]. First, image database, evaluation method is explained in section 3.1. Next, evaluation results are shown in section 3.2.

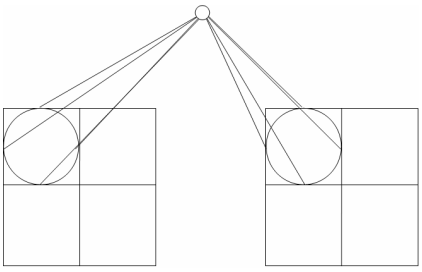


Figure 2: Reconstruction by local KPCA

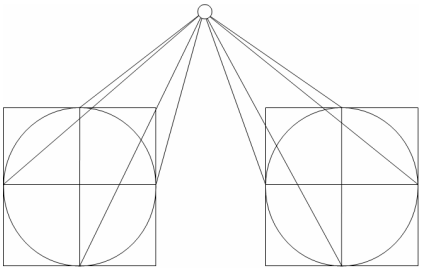


Figure 3: Reconstruction by global KPCA

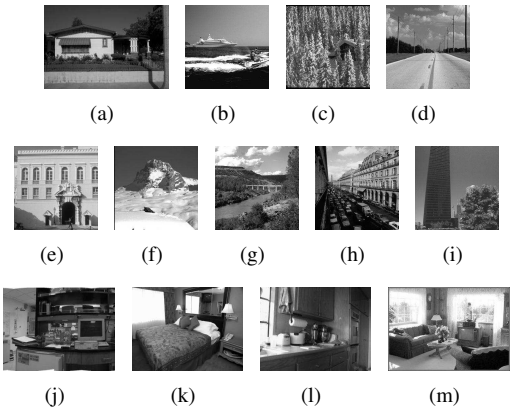


Figure 4: Examples of 13 scene images. (a) suburb (b) coast (c) forest (d) highway (e) inside-city (f) mountain (g) open-country (h) street (i) tall-building (j) office (k) bedroom (l) kitchen (m) living-room

3.1 Image database and evaluation method

We use the database of 13 scene categories in order to compare our method with conventional studies [11, 12, 2, 13, 14]. The database includes only gray-level images with various sizes. Each scene category has different number of images. Examples of 13 scene categories are shown in Figure 4. There are various scene categories such as outdoor and indoor. The within-class variance in scene classification is larger than that in face recognition problem because camera angle and objects in images are not static.

In this paper, the images of each scene category are divided into two sets; training and test sets. 100 images selected randomly are used as training set. The remaining images of each scene category are used as test set. This protocol is the same as [11, 12, 2, 13, 14].

Each scene category has the different number of test images. The minimum and maximum number of test image of a class is 110 and 310. To reduce the bias of different number of test images, the mean of the classification rate of each scene category is used in evaluation. This is also the same as conventional methods. We repeat this evaluation 3 times with different initial seed of a random function, and the mean classification rate of 3 runs is used as a final result.

3.2 Evaluation results

First, the proposed integration method is evaluated while changing the weight γ in equation (13). Figure 5 shows the result in which horizontal axis is γ and the vertical axis is the correct classification rate. Note that $\gamma = 0$ means the use of only global KPCA and $\gamma = 1$ means the use of only local KPCA. The 3 lines in the Figure mean the results with 3 different initial seeds for random function. The average classification rate of 3 runs is shown in Figure 6. Figures demonstrates that the integration of 2 KPCAs with different role improves accuracy. The best accuracy achieves more than 82.5% though the accuracy of only local KPCA or global KPCA is below 77%. Namely, about 6% in accuracy is improved by very simple weighted integration.

Table 1 shows that best accuracy of the proposed method, the accuracy of only local and global KPCA. The best accuracy of the weighted integration method is obtained at $\gamma = 0.79$. To show the baseline accuracy, we also evaluate the linear SVM of bag-of-visual words [27] which are commonly used in scene classification and object categorization. The basic feature for computing the visual words is 4 orientation histograms on 2×2 grid which are same as the proposed method. Table 1 also shows the accuracy. It achieves below 73%. This result shows the effectiveness of our method.

Finally, our method is compared with the conventional methods using the same database [11, 12, 13, 14, 2]. In general, the classification accuracy depends on the features and classifiers. Since each conventional method used different features and classifiers, the direct comparison with our method is difficult. Comparison result is shown Table 2. Note that accuracy of conventional methods is obtained from each paper. Since two methods [11, 12] used the bag-of-visual words with the local parts obtained from evenly sampled grid, they are similar with linear SVM of bag-of-visual words implemented by us. In [13], orientation histograms were developed from subregions with various sizes. Our simple approach gives much better accuracy than the method. In [14], auto-correlation in KPCA space of visual words is used to give shift-invariance and relative information with neighboring regions to feature. The proposed method integrates the shift-invariance similarity by lo-

Table 1: Evaluation result

Method	Classification rate
Proposed method	**82.63%**
local KPCA	76.62%
global KPCA	74.71%
linear SVM of bag-of-words	72.66%

Table 2: Comparison with conventional methods

Method	Classification rate
Proposed method	**82.63%**
[2] (PAMI2008)	85.9%
[28] (ICPR2010)	84.33%
[14] (ICIP2009)	81.43%
[13] (ICVS2008)	76.12%
[12] (ECCV2006)	73.4%
[11] (CVPR2005)	65.2%

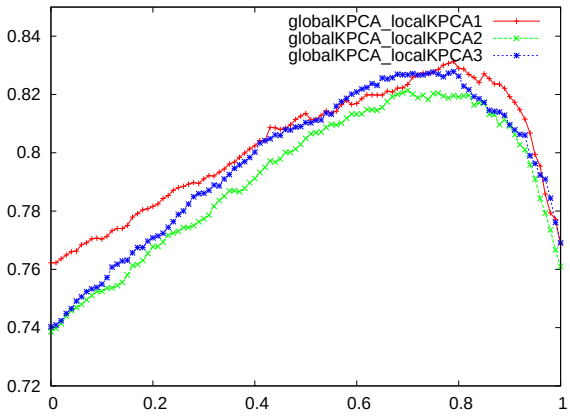


Figure 5: Accuracy of the proposed integration method

cal KPCA and the similarity with position dependent rough composition by global KPCA. Our simple approach outperforms the result in [14]. Unfortunately, our method is worse than the method [2] using spatial pyramid probabilistic Latent Semantic Analysis and the method [28] using local co-occurrence features. However, those methods used many devices while the proposed method is very simple in which the reconstruction errors of 2 KPCAs are integrated by only one parameter. In addition, the simple integration method is comparable to conventional methods though the direct comparison is difficult because of different features and classifiers. This shows the possibility of our approach. The accuracy will be improved further if we extend the proposed approach. This is a subject of future works.

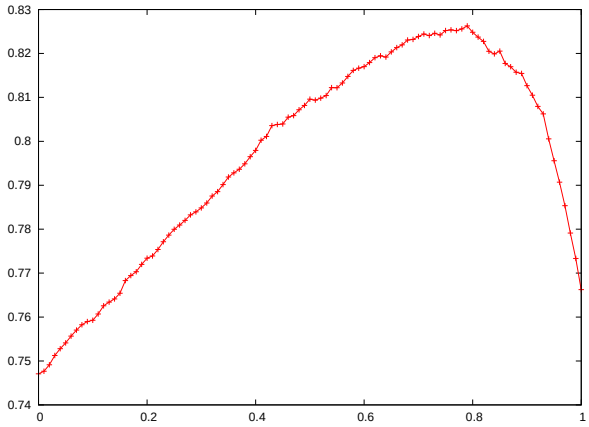


Figure 6: Average accuracy of 3runs

4 CONCLUSIONS AND FUTURE WORKS

We proposed a scene classification method using the integration of rough composition by global KPCA and fine part by local KPCA. By giving the obvious different role to both KPCAs, the simple weighted integration improved about 6% in comparison with only local and global KPCAs. The proposed method also outperformed the linear SVM of bag-of-visual words with the same features. Our very simple approach gave comparable accuracy with conventional methods using the same database. This shows the possibility of our approach.

In this paper, the simple weighted integration is used, and the accuracy is evaluated with the fixed weight for all test samples. However, if we select the appropriate weight for each test sample, the accuracy will be improved further. We may use the particle filter to select the weight such as [4]. This is a subject for future works.

REFERENCES

[1] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision* **60**(2), pp. 91–110, 2004.

[2] A. Bosch, A. Zisserman, and X. Munoz, "Scene classification using a hybrid generative/discriminative approach," *IEEE Trans. Pattern Analysis and Machine Intelligence* **30**(4), pp. 712–727, 2008.

[3] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *Proc. CVPR Workshop of Generative Model Based Vision*, 2004.

[4] K. Hotta, "Pose independent classifcation from small number of training samples based on kernel principal component analysis of local parts," *Im-*

age and Vision Computing **27**(9), pp. 1240–1251, 2009.

[5] A. Li, S. Shan, X. Chen, and W. Gao, "Maxmizing intra-indivisual correlations for face recognition across pose differences," in *Proc. IEEE CS Conference on Computer Vision and Pattern Recognition*, 2009.

[6] K. Hotta, "Local normalized linear summation kernel for fast and robust recognition," *Pattern Recognition* **43**(3), pp. 906–913, 2010.

[7] K. Hotta, "View independent face detection based on horizontal rectangular features and accuracy improvement using combination kernel of various sizes," *Pattern Recognition* **42**(3), pp. 437–444, 2009.

[8] R. P. N. Rao and D. H. Ballard, "Efficient encoding of natural time varying images produces oriented space-time receptive fields," tech. rep., 97.4, Dept of Comp Sci, Univ of Rochester, 1997.

[9] K. Murphy, A. Torralba, D. Eaton, and W. Freeman, "Object detection and localization using local and global features," in *Toward Category-Level Object Recognition*, pp. 382–400, 2006.

[10] T. Ishihara, K. Hotta, and H. Takahashi, "Estimation of object position based on color and shape contextual information," in *Proc. International Conference on Image Analysis and Processing, LNCS Vol.5716*, pp. 57–62, 2009.

[11] L. Fei-Fei and P. Perona, "A bayesian hierarchical model for learning natural scene categories," in *Proc. IEEE CS Conference on Computer Vision and Pattern Recognition*, pp. 524–531, 2005.

[12] A. Bosch, A. Zisserman, and X. Munoz, "Scene classification via plsa," in *Proc. 9th European Conference on Computer Vision*, pp. 517–530, 2006.

[13] K. Hotta, "Scene classification based on multi-resolution orientation histogram of gabor features," in *Proc. International Conference on Computer Vision Systems, LNCS Vol.5008*, pp. 291–301, 2008.

[14] K. Hotta, "Scene classification based on local autocorrelation of similarities with subspaces," in *Proc. IEEE International Conference on Image Processing*, pp. 2053–2056, 2009.

[15] *13 Scene categories database*. http://vision.cs.princeton.edu/ Databsets/SceneClass13.rar.

[16] K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf, "An introduction to kernel-based learning algorithms," *IEEE Trans. Neural Networks* **12**(2), pp. 181–201, 2001.

[17] B. Schölkopf, C. Burges, and A. Smola, *Advances in kernel methods: support vector learning*, MIT Press, 1998.

[18] R. Debnath and H. Takahashi, "Kernel selection for the support vector machine," *IEICE Trans. Info. & Syst.* **E87-D**(12), pp. 2903–2904, 2004.

[19] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*, Cambridge University Press, 2004.

[20] K. Hotta, "Non-linear feature extraction by linear principal component analysis using local kernel," *Pattern Recognition Recent Advances* , pp. 99–109, 2010.

[21] T. Balachander and R. Kothari, "Kernel based subspace pattern classification," in *Proc. International Joint Conference on Neural Networks*, vol. 5, pp. 3119–3122, 1999.

[22] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press Ltd., 1983.

[23] S. Watanabe, *Knowing and Guessing - Quantitative Study of Inference and Information*, John Wiley & Sons, 1969.

[24] S. Watanabe and N.Pakvasa, "Subspace method of pattern recognition," in *Proc. 1st International Joint Conference on Pattern Recognition*, pp. 25–32, 1973.

[25] B. Moghaddam and A. Pentland, "Probabilistic visual learning for object representation," *IEEE Trans. Pattern Analysis and Machine Intelligence* **19**(7), pp. 696–710, 1997.

[26] K. Hotta, "Robust face recognition under partial occlusion based on support vector machine with local gaussian summation kernel," *Image and Vision Computing* **26**(11), pp. 1490–1498, 2008.

[27] G. Csurka, C. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categoraization with bags of keypoints," in *Proc. ECCV Workshop on Statistical Learning in Computer Vision*, pp. 1–16, 2004.

[28] K. Hotta, "Scene classification using local co-occurrence feature in subspace obtained by kpca of local blob visual words," in *Proc. International Conference on Pattern Recognition*, pp. 4230–4233, 2010.

Gait Recognition in the Presence of Occlusion: A New Dataset and Baseline Algorithms

Martin Hofmann[1], Shamik Sural[2], Gerhard Rigoll[1]

[1] Institute for Human-Machine Communication, Technische Universität München, Germany
{martin.hofmann,rigoll}@tum.de
[2] Indian Institue of Technology Kharagpur, India
shamik.sural@gmail.com

ABSTRACT

Human gait is an important biometric feature for identification of people. In this paper we present a new dataset for gait recognition. The presented database overcomes a crucial limitation of other state-of-the-art gait recognition databases. More specifically this database addresses the problem of dynamic and static inter object occlusion. Furthermore this dataset offers three new kinds of gait variations, which allow for challenging evaluation of recognition algorithms. In addition to presenting the database we present two baseline algorithms (Color histograms, Gait Energy Image) to perform person identification using gait. These algorithms already show promising results on the presented database.

Keywords: biometrics, gait recognition, database, occlusion, gait energy image.

1 INTRODUCTION

Person identification by biometric features is a well established research area. The main focus has so far been on physiologic features such as face, iris and fingerprint. In addition, behavior based features such as voice, signature and gait can be used for person identification. In this work we contribute to the research of person identification using gait. The main advantage of using these features over other physiologic features is the possibility to identify people from large distances and without the person's direct cooperation. For example, in low resolution images, a person's gait signature can be extracted, while the face is not even visible. Also no direct interaction with a sensing device is necessary, which allows for undisclosed identification. Thus gait recognition has great potential in video surveillance, tracking and monitoring.

Studies suggest [13] that if all gait movements are considered, gait is unique. These findings are the basis of the assumption that recognition using only gait must also be possible for a computer system. Over the last decade the field of recognizing people using gait features has received remarkable attention. A multitude of methods and techniques in feature extraction as well as in classification have been developed. Experiments are promising and encouraging.

While good datasets for training and evaluation are available (See summary in Section 3), we find that all of them ignore to address one important challenge: The challenge of occlusions. Occlusions are annoying but are unfortunately omnipresent in practice. Especially in a real word surveillance scenario, occlusions occur frequently. Typical gait recognition algorithms require a full gait cycle[1] for recognition. In the case of occlusion, however, it becomes a challenging problem to extract a full gait cycle. In heavy occlusion, parts of the gait cycle might be visible, while other parts are obscured by another person walking in front. The challenge then lies in stitching together parts of different gait cycles in order to obtain one complete gait cycle. Alternatively gait recognition algorithms could be developed for which parts of the gait cycle are sufficient. While to date, no algorithm is capable of handling partially observable gait cycles, we here present the TUM-IITKGP gait dataset, which can be used to specifically address occlusions.

To this end the presented database includes recordings with two kinds of occlusions. On the one hand dynamic occlusions by people walking in the line of sight of the camera and on the other hand static occlusions by people who are occluding the person of interest by standing in the scene. In addition to specifically addressing the occlusion challenge, the TUM-IITKGP dataset also features three new configuration variations, which allows to test algorithms for their capability of handling changes in appearance.

We present two baseline algorithms for recognition on this dataset. The first algorithm uses appearance

[1] A full gait cycle is the time interval between successive instances of initial foot-to-floor contact for the same foot

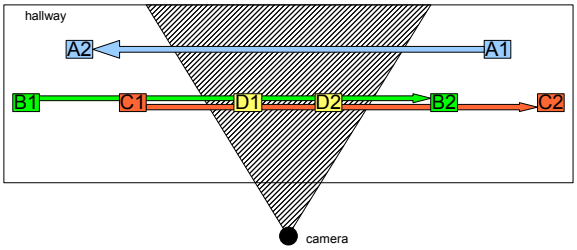


Figure 1: Physical setup of the recording

information based on color histograms. Thus this algorithm is not precisely a gait recognition algorithm, but shows promising results. The second algorithm is an actual gait recognition algorithm based on the well known Gait Energy Image (GEI) features [6][7]. It can be seen that this baseline algorithm which uses only motion information and no color information already shows excellent results.

Section 2 gives a summary of related gait recognition databases. Section 3 presents the new dataset in details. We then present in Section 4.1 a simple baseline recognition system based on color histograms, as well as an actual gait recognition baseline algorithm in Section 4.2. Results are given in Section 6 and we conclude in Section 7.

2 RELATED GAIT DATABASES

Since the field of gait recognition has been in existence for roughly a decade, the research community has long utilized publicly available databases for comparative performance evaluation.

Table 1 summarizes the most prominent gait recognition corpora. This table also shows the important features of the particular databases. The most important features of a database are the number of subjects (which should be high), as well as a good set of person variations. These variations include, but are not limited to, view angle, clothing, shoe types, surface types, carrying condition, illumination, and time.

The first available dataset was the 1998 UCSD Dataset [11], which contains merely 6 subjects. Most of the following early gait recognition databases were published in 2001 from various institutions [2][3][5][9][10][12]. Those datasets feature a medium number (about 25) of subjects. It was then found, that for meaningful evaluation, datasets should contain at least 30 subjects and possibly more.

The most comprehensive database to date, which features a large set of subjects as well as a substantial set of variations is probably the HumanID Gait Challenge [15]. Other databases such as CASIA (Dataset B) [1] also feature high numbers of subjects and a significant number of variations. CASIA additionally features an exhaustive number of views, which allows for precise 3D reconstruction.

3 THE TUM-IITKGP DATABASE

As established in the introduction, the rationale behind recording a new gait recognition dataset is to specifically address the problem of occlusions, which would frequently occur in real world applications. The TUM-IITKGP Database currently consists of 840 sequences from 35 individuals.

The physical setup can be seen in Figure 1. The camera is set up in a rather narrow hallway, reflecting a realistic setup in a potential real world surveillance application. The camera is positioned at a medium height of 1.85 meters and is oriented perpendicular to the hallway direction. Thus people are walking from right to left and from left to right in the image.

Each person is captured in six different configurations. Furthermore, each of the configurations is repeated two times in a right-to-left motion and two times in a left-to-right motion, resulting in a total of 840 sequences. Table 2 and Figure 2 show the six configurations for each person. Each person was primarily recorded in a *regular* walking configuration, followed by three degenerated configurations including hands in *pocket*, *backpack* and *gown*. These configurations can be used to evaluate recognition methods in the presence of different kinds of gait variation.

Furthermore two configurations are specifically designed to evaluate performance in the presence of occlusions. One is with two people walking past (*dynamic occlusion*). The other is with two people just standing in the line of sight (*static occlusion*).

In all of the six recordings, the person of interest (the subject) is starting to walk at point A1 and ending at point A2. In case of *dynamic occlusions* (configuration 5), the two other people are walking from B1/C1 to B2/C2, respectively. For *static occlusions* (configuration 6), the two additional people are standing at D1 and D2, respectively.

Overall, each configuration is repeated 4 times. For the second iteration the walking directions are inversed. Thus the subject is walking back from A2 to A1, and in case of occlusion configuration 5, the occluding people are also walking in the opposite direction. The third iteration is equivalent to the first and the fourth is equivalent to the second.

	Short Name	Description
Conf. 1	regular	Regular walking
Conf. 2	pocket	Walk with hands in pocket
Conf. 3	backpack	Walk with a backpack
Conf. 4	gown	Walk with gown
Conf. 5	dynamic occlusion	Occlusion by two walking people
Conf. 6	static occlusion	Occlusion by two standing people

Table 2: Walking configurations

Database, Ref.	#subjs.	#seqs.	Environment	Time	Variations
UCSD ID [11]	6	42	Outdoor, Wall background	1998	Time (minutes)
CMU Mobo [5]	25	600	Indoor, Tredmill	2001	Viewpoint, Walking speeds, Carrying conditions, Surface incline
Georgia Tech [9]	15	268	Outdoor	2001	Time(6 months), viewpoint
	18	20	Magnetic tracker	2001	Time(6 months)
HID-UMD Dataset 1 [10]	25	100	Outdoor	2001	
HID-UMD Dataset 2 [3]	55	222	Outside, Top mounted	2001	viewpoints (front, side), time
MIT, 2001 [2]	24	194	Indoor	2001	view, time (minutes)
Soton Small Database [12]	12	-	Indoor, green background	2001	carrying condition, clothing, shoe, view
Soton Large Database [12]	115	2128	Indoor, Outdoor, Tredmill	Summer 2001	view
HumanID Gait Challenge [15]	122	1870	Outdoor	May & Nov. 2002	viewpoint, surface, shoe, carrying condition, time (months)
CASIA Database A [1]	20	240	Outdoor	Dec. 2001	3 viewpoints
CASIA Database B [1]	124	13640	Indoor	Jan 2005	11 viewpoints, clothing, carrying condition
CASIA Database C [1]	153	1530	Outdoor, night, thermal camera	2005	speed, carrying condition
TUM-IITKGP	35	840	Indoor, Hallway, Occlusions	Apr. 2010	time (minutes), carrying condition, occlusions

Table 1: Comparison of other gait recognition databases

(a) regular (b) pocket (c) backpack

(d) gown (e) dynamic occlusion (f) static occlusion

Figure 2: Example images from all six configurations

4 BASELINE ALGORITHMS

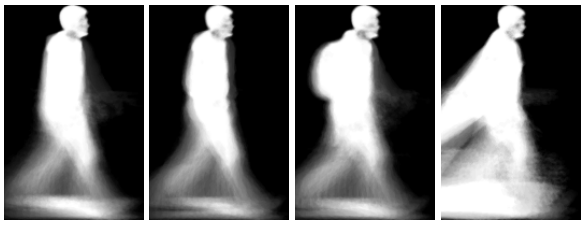
In order to show first recognition results and in order to have a means of comparing other algorithms for future performance evaluation, we applied two baseline algorithms to the database.

Both methods are non-model based. The first method uses color histograms for feature extraction, the second method uses Gait Energy Image (GEI) [7]. Obviously using only color information has a multitude of drawbacks, most importantly the fact that this kind of feature is not invariant to change of clothing.

The second method however is a true gait recognition method, because the Gait Energy Image captures temporal motion over a gait cycle and is independent from any appearance based features such as color.

4.1 Baseline Algorithm using Color Histograms

Using color histograms is a widely used technique for recognition and re-identification of people. This holds especially true for short-time recognition, where people do not change their appearance and clothing. Color histograms are extremely fast and easy to compute. Furthermore no detection of body parts is necessary, because the feature can be extracted globally from the full person. Besides the problem that color features fail in case of change in clothing, another drawback is that they are very sensitive to lighting differences especially when recognition is to be performed between differently calibrated cameras. This however can be handled using adaptive appearance transformations such as the Brightness Transfer Function [14].For this work, we use 4-by-4-by-4 3D color histograms H. Thus each person in the database is represented by a 4096 dimensional sparse feature vector. To extract this feature vector, we first use background modeling based on Gaussian Mixture Models [16] to segment foreground blobs. The color histograms are then computed over all foreground segments on the full sequence.For recognition we use nearest neighbor classification, where H_j is the



(a) regular (b) pocket (c) backpack (d) gown

Figure 3: Gait Energy Images for four configurations

j-th sample from the test set, and $\overline{H}_i$ is the mean of the samples in the i-th class from the training set.

$$L_j = \underset{i}{\operatorname{argmin}} d_X(H_j, \overline{H}_i) \tag{1}$$

Here $d_X = \{d_{euclid}, d_{corr}, d_{bhatt}, d_{chi}\}$ is the distance function of one of four different histogram comparison measures: Euclidean distance, normalized correlation, Bhattacharyya distance and Chi Squared distance with the following respective formulas:

$$d_{euclid} = \sqrt{\sum_{Bins} (H_1 - H_2)^2} \tag{2}$$

$$d_{corr} = 1 - \frac{\sum_{Bins}(H_1 - \bar{H}_1)(H_2 - \bar{H}_2)}{\sum_{Bins}\sqrt{(H_1 - \bar{H}_1)^2 (H_2 - \bar{H}_2)^2}} \tag{3}$$

$$d_{bhatt} = 1 - \sum_{Bins} \sqrt{\frac{H_1}{|H_1|}\frac{H_2}{|H_2|}} \tag{4}$$

$$d_{chi} = \sum_{Bins, H_1 + H_2 \neq 0} \frac{(H_1 - H_2)^2}{H_1 + H_2} \tag{5}$$

In the experiments it turned out that all four of these distance measures performed similarly well with a slight tendency of the Chi Squared distance being the best. See results in Section 6.

4.2 Baseline Algorithm based on Gait Energy Image

In contrast to the color histogram method presented in the previous section, GEI [6] is considered a true gait recognition method, because the used features only make use of silhouette and motion information. Appearance and color information is discarded.

4.3 Feature Extraction using GEI

In essence, the Gait Energy Image is an arithmetic mean of the binarized foreground blobs. Denote B_t the foreground silhouette in frame t. Then, the Gait Energy Image g is formally defined as the silhouette average over one full gait cycle:

$$g(x,y) = \frac{1}{T}\sum_{t=1}^{T} B_t(x,y) \tag{6}$$

Here, T is the number of frames in a full gait cycle. Using this kind of feature greatly reduces the available data, since all the gait information is compressed to only one gray level image. Figure 3 shows Gait Energy Images for the first four configurations. It has been shown that this representation suffices for person identification [7].

4.4 Feature Space Reduction

The gait energy images $g(x,y)$ have a resolution of 130×200 pixels. Thus the feature vector is still large with 26000 dimensions. We apply principal component analysis (PCA) followed by multiple discriminant analysis (MDA) to reduce the size of the feature vector. A combination of PCA and MDA, as proposed in [8], results in the best recognition performance. While PCA seeks a projection that best represents the data [4], MDA seeks a projection that best separates the data [8].

Assume that the training set, consisting of N d-dimensional training vectors $\{g_1, g_2, \ldots, g_N\}$, is given. Then the projection to the $d' < d$ dimensional PCA space is given by

$$y_k = U_{pca}(g_k - \overline{g}), \qquad k = 1, \ldots, N \tag{7}$$

Here U_{pca} is the $d' \times d$ transformation matrix with the first d' orthonormal basis vectors obtained using PCA on the training set $\{g_1, g_2, \ldots, g_N\}$ and $\overline{g} = \sum_{k=1}^{N} g_k$ is the mean of the training set. After PCA, MDA is performed. It is assumed that the reduced vectors $\mathscr{Y} = \{y_1, y_2, \ldots, y_N\}$ belong to c classes. Thus the set of reduced training vectors $\mathscr{Y}$ is composed of its c disjunct subsets $\mathscr{Y} = \mathscr{Y}_1 \cap \mathscr{Y}_2 \cap \ldots \mathscr{Y}_c$. The MDA projection has by construction $(c-1)$ dimensions. These $(c-1)$ dimensional vectors z_k are obtained as follows

$$z_k = V_{mda} y_k, \qquad k = 1, \ldots, N \tag{8}$$

where V_{mda} is the transformation matrix obtained using MDA. This matrix results from optimizing the ratio of the between-class scatter matrix S_B and the within-class scatter matrix S_W:

$$J(V) = \frac{|\widetilde{S}_B|}{|\widetilde{S}_W|} = \frac{|V^T S_B V|}{|V^T S_W V|}. \tag{9}$$

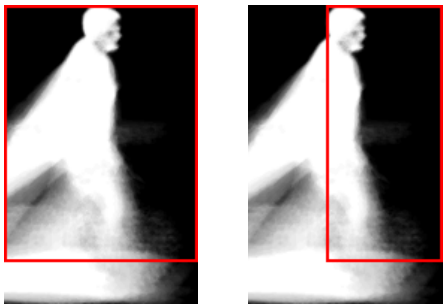
Here the within-class scatter matrix S_W is defined as $S_W = \sum_{i=1}^{c} S_i$, with $S_i = \sum_{y \in \mathscr{Y}_i} (y - m_i)(y - m_i)^T$ and $m_i = \frac{1}{N_i}\sum_{y \in \mathscr{Y}_i} y$. Where $N_i = |\mathscr{Y}_i|$ is the number of vectors in $\mathscr{Y}_i$. The between-class scatter S_B is defined as $S_B = \sum_{i=1}^{c} N_i (m_i - m)(m_i - m)^T$, with $m = \frac{1}{N}\sum_{i=1}^{c} N_i m_i$.

Finally, for each Gait Energy Image, the corresponding gait feature vector is computed as follows

$$z_k = U_{pca} V_{mda}(g_k - \overline{g}) = T(g_k - \overline{g}), \qquad k = 1, \ldots, N \tag{10}$$

4.5 Classification

For further classification, we use nearest neighbor classification on this reduced set of feature vectors. To this



(a) Baseline 2a (b) Baseline 2b

Figure 4: Cropped GEI regions used for recognition

end, first the mean feature vector $\bar{z}_i$ is calculated for each class.

$$\bar{z}_i = \frac{1}{|\mathscr{Z}_i|} \sum_{z \in \mathscr{Z}_i} z. \tag{11}$$

For each Gait Energy Image from the test set $\hat{g}_j$, we perform the identical transformation to get the reduced feature vector

$$\hat{z}_j = T(\hat{g}_j - \bar{g}) \tag{12}$$

Person identification then becomes a nearest-neighbor classification. We assign a class label L_j to each test gait image according to

$$L_j = \underset{i}{\operatorname{argmin}} ||\hat{z}_j - \bar{z}_i|| \tag{13}$$

4.6 Implementation details

Besides the principle approach as it was described above, there are several technical details that had to be considered. First, for our experiments, we align the foreground blobs B_t before calculating the GEI. This is done by centering each blob B_t based on the centroid of the top 10% of each blob. This way it is guaranteed that the heads, which are most stable in recognition, are all aligned at the same position.

Second, we found (just like others have [7] [15]), that using the full Gait Energy Images for recognition does not result in the best performance. Especially the lower region of the image is quite troublesome, because of shadows and reflections on the ground, as well as different floor types (as in [15]). Therefore we decided to use only the top 80% of the GEIs. Figure 4 depicts the cropping regions.

In addition we experimented with a second cropped variation of the GEIs. Here we use the top 80% of the image, and only the rightmost 60% of the image. This way, only the frontal part of the persons are included. This is beneficial, because this way the gown and the backpack have a much smaller impact on the Gait Energy Images. In Section 6 we show that this cropping indeed leads to improved recognition rates.

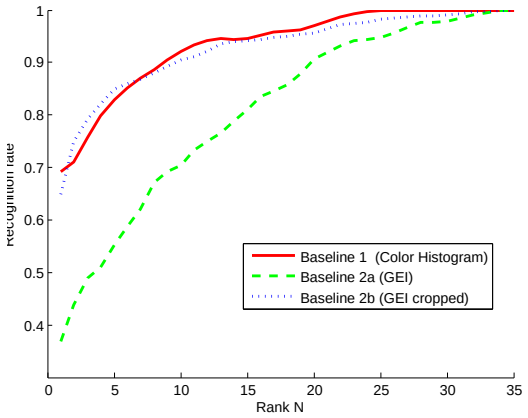


Figure 5: Rank N recognition rates for the three Baseline variants

5 EVALUATION METHOD

The presented gait recognition database is meant to be a basis for performance evaluation of various present and future gait recognition algorithms. Therefore, evaluation should be carried out the same way for all algorithms. We propose the following procedure:

The goal is to recognize a person, which has only been seen once before. Thus the training set consists of merely one single sequence of each person. We define that this sequence is one from the first configuration (regular walking). Consequently the test set consists of 23 sequences for each person (three for *regular* and four each for the other five configurations). Because the database consists of 35 people, the overall test set consists in total of 805 sequences.

Since there are four sequences for the first configuration, a 4-fold cross validation is performed. This means that there are 4 rounds of evaluation, each time with one of the four sequence in regular configuration as the sole training sample, and all the rest as the test. The result is then averaged over the 4 rounds of evaluation.

6 RESULTS

We evaluate on the one hand the color histogram based recognition method (Baseline 1), described in Section 4.1 and on the other hand we evaluate GEI approach (Baseline 2) described in Section 4.2. In the second case we evaluate the recognition rate for the two differently cropped Gait Energy Images. However, at this point, the Gait Energy Image could not be calculated for configuration 5 and 6, thus there are no results in those cases.

Results are shown in Table 3. We evaluate the recognition rate for each of the configurations separately and we report overall recognition rates. For all three variants, we report rank 1, rank 5 and rank 10 recognition rates. Figure 5 additionally shows the cumulative matching characteristic (CMC) for the rank n recognition rates for all three baseline variant.

	Top 1			Top 5			Top 10		
	BL 1	BL 2a	BL 2b	BL 1	BL 2a	BL 2b	BL 1	BL 2a	BL 2b
Conf. 1	97.9%	68.6%	77.1%	100%	76.2%	94.3%	100%	85.7%	97.1%
Conf. 2	93.3%	67.1%	75.7%	93.3%	80.7%	94.3%	100%	90.0%	97.8%
Conf. 3	75.0%	11.4%	77.1%	91.7%	45.7%	90.0%	100%	66.4%	94.3%
Conf. 4	20.0%	8.6%	32.9%	60.0%	23.6%	63.6%	73.3%	43.5%	74.3%
All(1-4)	69.9%	36.9%	64.9%	85.2%	55.2%	84.9%	92.6%	70.5%	90.5%
Conf. 5	43.7%	-	-	60.4%	-	-	77.1%	-	-
Conf. 6	70.0%	-	-	90%	-	-	100%	-	-
All(1-6)	65.8%	-	-	81.9%	-	-	91.1%	-	-

Table 3: Results for Baseline 1 (Color Histogram), Baseline 2a (Gait Energy Image) and Baseline 2b (Cropped Gait Energy Image)

It can be seen that the simple color based recognition method outperforms the GEI approach. However, the GEI approach also shows excellent results, and in case of the cropped GEI, the performance of GEI surpassed the performance of the color histogram method.

7 CONCLUSIONS

In this paper we have presented a new gait recognition database, which is focused on the problem of occlusions. Besides addressing the occlusion problem, the database also addresses three new kinds of variations which have not yet been addressed by other datasets. More specifically these variations include hands in *pocket*, wearing *backpack* and waring a *gown*.

We have presented two baseline algorithms which perform excellent on the given dataset for the case of no occlusion. However, so far neither of these two algorithms specifically addresses the occlusion problem, resulting in low performance for those cases. Thus it remains future work of actually utilizing the databases capabilities to show good performance in spite of occlusions.

8 ACKNOWLEDGMENTS

This work has been partially funded by the European Projects FP-214901 (PROMETHEUS) as well as by Alexander von Humboldt Fellowship for experienced researchers.

REFERENCES

[1] Center for biometrics and security research, CASIA. http://www.cbsr.ia.ac.cn.

[2] R. T. Collins, R. Gross, and J. Shi. Silhouette-based human identification from body shape and gait. In *Proceedings of IEEE Conference on Face and Gesture Recognition*, pages 351–356, 2002.

[3] N. Cuntoor, A. Kale, and R. Chellappa. Combining multiple evidences for gait recognition. In *Proc. ICASSP*, pages 6–10, 2003.

[4] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley, New York, 2. edition, 2001.

[5] R. Gross and J. Shi. The cmu motion of body (mobo) database. Technical report, 2001.

[6] J. Han and B. Bhanu. Statistical feature fusion for gait-based human recognition. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:842–847, 2004.

[7] J. Han and B. Bhanu. Individual recognition using gait energy image. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(2):316–322, 2006.

[8] P. Huang, C. Harris, and M. Nixon. Recognising humans by gait via parametric canonical space. *Journal of Artificial Intelligence in Engineering*, 13(4):359–366, November 1999.

[9] A. Y. Johnson and A. F. Bobick. A multi-view method for gait recognition using static body parameters. In *Proceedings of the Third International Conference on Audio- and Video-Based Biometric Person Authentication*, pages 301–311, 2001.

[10] A. Kale, N. Cuntoor, and R. Chellappa. A framework for activity-specific human identification. In *International Conference on Acoustics, Speech and Signal Processing*, 2002.

[11] J. Little and J. E. Boyd. Recognizing people by their gait: The shape of motion. *Videre*, 1:1–32, 1996.

[12] J. S. M. Nixon, J. Carter and M. Grant. Experimental plan for automatic gait recognition. Technical Report 2001.

[13] M. P. Murray, A. B. Drought, and R. C. Kory. Walking patterns of normal men. *Journal of Bone and Joint Surgery*, 46-A(2):335–360, 1964.

[14] B. Prosser, S. Gong, and T. Xiang. Multi-camera matching using bi-directional cumulative brightness transfer functions. In *British Machine Vision Conference*, 2008.

[15] S. Sarkar, P. J. Phillips, Z. Liu, I. R. Vega, P. Grother, and K. W. Bowyer. The humanID gait challenge problem: Data sets, performance, and analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27:162–177, 2005.

[16] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, 2:2246, 1999.

Improved Algorithm for Principal Curvature Estimation in Point Clouds due to Optimized Osculating Circle Fitting based on Geometric Algebra

Roman Getto
TU Darmstadt,Germany
Departement of Computer Science
Germany 64283, Darmstadt
r_getto@rbg.informatik.tu-darmstadt.de

Dietmar Hildenbrand
TU Darmstadt,Germany
Departement of Computer Science
Germany 64283, Darmstadt
Dietmar.hildenbrand@gris.informatik.tu-darmstadt.de

ABSTRACT

In this paper we introduce our improvements and innovations to an algorithm for the estimation of principal curvatures in point clouds. The major part of the improvement is achieved by the use of a new osculating circle fitting. In this paper, first of all, we explain the algorithm for Principal Curvature Estimation, as well as the previous osculating circle fitting. Then we present the new osculating circle fitting and an additional possibility of improving the results by a method, which adjusts some variables of the algorithm at runtime, to adapt the algorithm to the denseness of the particular point cloud. Furthermore, we present some results of the new methods. To complete this paper, we give a conclusion and an outlook.

Keywords

Geometric Algebra, Point Sets, Point Clouds, Principal Curvature, Osculating Circle Fitting

1. INTRODUCTION

With a 3D scanner, one can construct a digital model of a real object. However, to get a model most close to reality as possible, a couple of steps are needed. The 3D scanner produces a point cloud, mostly accurate, but, depending on the physical characteristics of the object, containing some small measurement errors. If the digital model is simply constructed by combining three neighbored points to a triangle, the measurement errors leads to several undesirable effects e.g., under certain conditions, a plane part of the surface can look cragged in the reconstructed digital model. To obtain mostly correct reconstructions of the surface, complex methods like presented in [Gun08], [Hor06], [Med05], [Kol04] and [Ada03] are needed.

Another solution is to avoid the effects by reconstructing the object with the help of the principal curvatures of the surface, like presented in [Goi06]. The two principal curvatures are defined for each point of a surface and are the maximum and minimum curvature in a particular direction of the point. Therefore, these indicate how the surface is formed, e.g. a point on a plane has a maximum and minimum curvature of 0. If the two principal curvatures, the minimum and the maximum curvature, are not equal, then the directions in which they occur are clearly defined. The directions of the principal curvatures are the principal directions.

For such a solution, which uses the principal curvatures, first of all, the knowledge of the principal curvatures and directions for each point in the point cloud is needed. Unfortunately, the principal curvatures are not directly generated by the 3D scanner. We need to estimate them with the point cloud. Some approaches to this task are presented in [Kal07], [Yan06] and also in [Goi06].

A new approach to this task is presented in [Sei10]. The presented algorithm uses an osculating circle fitting, which is based on the geometric algebra, for estimating the curvature. The algorithm already leads to useful estimations of the principal curvatures and the corresponding principal directions for each point of a point cloud. Nevertheless, there is room for improvement. The accuracy of the estimation as well as the performance of the algorithm are of major interest. We present two innovations to this

algorithm. An improved osculating circle fitting, also based on the geometric algebra, and a newly developed method, which automatically adapts the algorithm to the structure of the point cloud.

In the second section we will explain the algorithm for principal curvature estimation. The previous osculating circle fitting will be explained separately to the algorithm, in the third section. In the following fourth section, the changed new version of the osculating circle fitting is presented and in the fifth section the new method, named dynamic variable adjustment, is introduced. Some results of the new methods are presented in the sixth section. At last, a conclusion and outlook is given in the seventh section.

2. PRINCIPAL CURVATURE ESTIMATION ALGORITHM

The algorithm described in this section is almost a summary of the algorithm presented in [Sei10]. The algorithm is explained in this part, since it is essential to understand the algorithm, for understanding the improvements and innovations.

The goal of the algorithm is to calculate the principal curvatures and the principal directions of a point $\boldsymbol{x}$, in a point cloud. To estimate the curvature, the Algorithm fits osculating circles to a set of points, like shown in Figure 1. The center $\boldsymbol{m}$ of the osculating circle is always on the straight line defined by $\boldsymbol{x}$ and his surface normal $\boldsymbol{n}$.

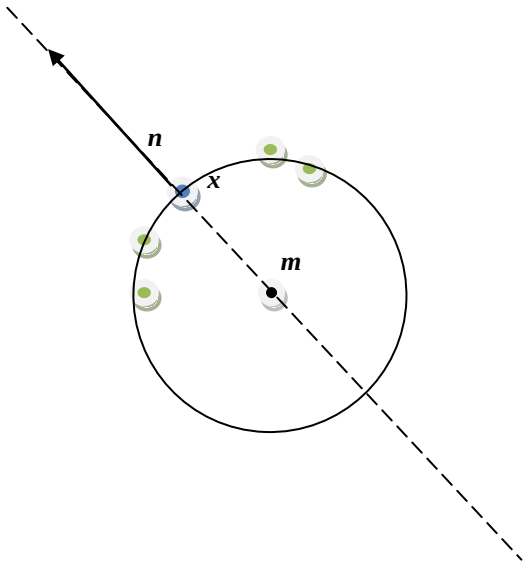


Figure 1. osculating circle for a set of points

The algorithm consists of 5 parts. The center of the algorithm, the third part, is the osculating circle fitting. The first and second part choose the points to which the osculating circles are fitted. The results of the fittings are used in the fourth and fifth part, to calculate the principal curvatures and principal directions.

Choice of the Surrounding Points

For estimating the principal curvatures of the arbitrary chosen point $\boldsymbol{x}$, only the surrounding points of $\boldsymbol{x}$ are needed. Therefore, the first step is to choose all points which distance to $\boldsymbol{x}$ is smaller than the value of a predefined surroundings-threshold. The choice of this value is very important as it decides how many points are used for the fittings.

Estimation of Planes in 16 Directions

An osculating circle fitting is made for each of the 16 directions. Therefore, we need 16 set of points which each represents one direction.

One set of points represents one direction when they are arranged one after another, nearly forming a two dimensional curve. For this reason, an osculating circle fitting makes sense. When we fit an osculating circle to the curve, the circle nearly has the same curvature as the curve and hence the same curvature as the surface in the corresponding direction of $\boldsymbol{x}$.

We achieve such a set of points if we calculate the intersection of the surface and a plane, containing the surface normal $\boldsymbol{n}$. Which direction the plane represents is defined by the second vector, with which the plane is defined. Therefore, we need to define vectors in 16 directions. Figure 2 shows how these 16 directions should be arranged. The green lines represent the 16 vectors of the 16 directions.

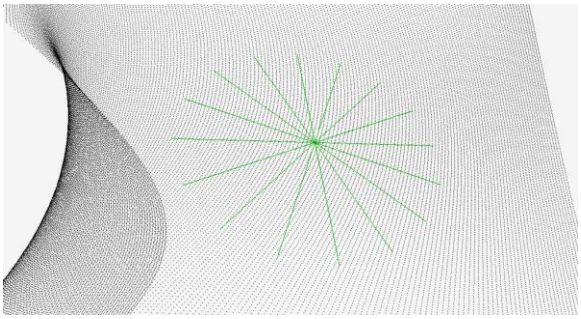


Figure 2. 16 directions for point x

Together with the surface normal each vector defines a plane. For a good representation of one direction each vector must be orthogonal to the surface normal. Therefore, we define the first vector $\boldsymbol{t_0}$ as follows:

$$\boldsymbol{n} = \begin{pmatrix} n_x \\ n_y \\ n_z \end{pmatrix} \quad \boldsymbol{t_0} = \begin{pmatrix} 1 \\ 1 \\ \dfrac{(-n_x - n_y)}{n_z} \end{pmatrix}$$

The remaining vectors $\boldsymbol{t_1}$up to $\boldsymbol{t_{15}}$ are calculated by repeatedly rotating $\boldsymbol{t_0}$ with $1/8\pi$ around $\boldsymbol{n}$, i.e. the vector $\boldsymbol{t_i}$ is the result of the rotation of $\boldsymbol{t_{i-1}}$ around $\boldsymbol{n}$, with the rotation angle $1/8\pi$.

The 16 planes defined by the surface normal and a corresponding $\boldsymbol{t_i}$actually are only "half" planes since they start at $\boldsymbol{x}$. Furthermore, the intersection of a plane and the surface cannot be perfectly calculated due to the fact that the point cloud is not infinite dense. For this reason, we define a plane-distance-threshold. Every point of the set of surrounding points, which distance to a plane is less than the plane-distance-threshold, is assigned to the set of points of the corresponding direction. As result we have 16 set of points which each represents a part of the surface in a certain direction.

Osculating Circle Fitting

For each of the 16 set of points an osculating circle fitting is made. How this fitting works, is explained in detail in the third section.

As result of the osculating circle fittings we get curvature values in 16 directions.

Sine Function Fitting

Since we do not know in which direction the principal curvatures are and also cannot be certain that one of the 16 sampled directions is exactly a direction of a principal curvature, we need to construct a function which interpolates the curvatures between the sampled directions.

Figure 3 shows the calculated curvatures in 16 directions for an arbitrary chosen point $\boldsymbol{x}$. It is no coincidence that the calculated curvatures are arranged like a sine function. Since the principal directions are orthogonal to each other, a sine function makes a good approximation to the curvatures in all directions.

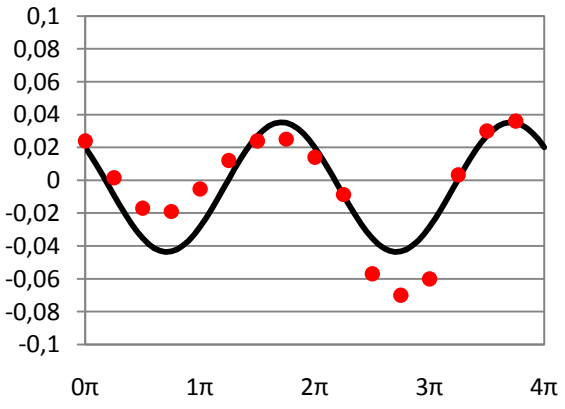


Figure 3. Fitted Sine Function

As we can see the sine function in Figure 3 does not perfectly approximate the curvature values. Since we estimate the curvature with "half" planes we fit 2 phases of the sine function. The correct curvature values are between the values for each phase. Therefore, the mean values represented by the sine function are accurate.

The fitted sine function has the form:

$$f(\alpha) = a \cdot \sin(\alpha + \varphi) + o$$

The variable a represents the amplitude, φ represents the phase and o represents the offset. The sine function fitting is based on the method described in [IEEE01].

For this method a matrix D and a vector $\boldsymbol{y}$ has to be defined, where y_i denotes the calculated curvature in the ith direction and α_i twice the rotation angle of the corresponding $\boldsymbol{t_i}$. It has to be twice the angle, because the 16 curvature values represent two phases of the sine function. E.g. $\boldsymbol{t_2}$ has the rotation angle $2/8\pi$, therefore α_2 is $4/8\pi$

$$D = \begin{pmatrix} \cos\alpha_0 & \sin\alpha_0 & 1 \\ \cos\alpha_1 & \sin\alpha_1 & 1 \\ \vdots & \vdots & \vdots \\ \cos\alpha_{15} & \sin\alpha_{15} & 1 \end{pmatrix} \qquad \boldsymbol{y} = \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{15} \end{pmatrix}$$

With D and $\boldsymbol{y}$ a result vector $\boldsymbol{r}$ can be calculated:

$$\boldsymbol{r} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \end{pmatrix} \qquad \boldsymbol{r} = (D_0^T D_0)^{-1}(D_0^T y)$$

Finally the amplitude, phase and offset of the fitted sine function can be calculated:

$$a = \sqrt{{r_1}^2 + {r_2}^2} \qquad o = r_3$$

$$\varphi = \begin{cases} \tan^{-1}\left(-\frac{r_2}{r_1}\right) + \frac{\pi}{2} & if\ r_1 > 0 \\ \tan^{-1}\left(-\frac{r_2}{r_1}\right) + \frac{3\pi}{2} & if\ r_1 < 0 \end{cases}$$

Calculation of the Principal Curvatures and Principal Directions

The principal curvatures are denoted as k_1 and k_2. k_1 is the minimal curvature and k_2 is the maximal curvature. Since the sine function represents the curvatures of $\boldsymbol{x}$ in all directions, the minimal and

maximal value of the sine function are exactly the value of k_1 and k_2:

$$k_1 = a - o \qquad k_2 = a + o$$

For the principal directions, we must know at which angle the extrema of the sine function are. A sine function without shifted phase would have his maximum at $\pi/2$ and his minimum at $3\pi/2$. We achieve the angle of the extrema if we subtract the phase from the normal extrema. Additionally, we must divide the angle by 2, since the sine function was fitted for values of two phases.

$$\gamma_1 = \frac{\frac{3\pi}{2} - \varphi}{2} \qquad \gamma_2 = \frac{\frac{\pi}{2} - \varphi}{2}$$

Finally, we achieve the vector of the direction of k_1, if we rotate $\boldsymbol{t_0}$ with γ_1 around $\boldsymbol{n}$. The rotation with γ_2 results in the vector of the direction of k_2.

3. PREVIOUS OSCULATING CIRCLE FITTING

The osculating circle fitting takes $\boldsymbol{x}$, $\boldsymbol{n}$ and a set of points as input, and calculates a circle, which is as near as possible at all points. The output of the osculating circle fitting is the radius r and the center $\boldsymbol{m}$ of the fitted circle.

A circle has the same curvature value at all his points: the multiplicative inverse of the radius. The circle is fitted to all points of the set and therefore nearly has the same curvature as the part of the surface which is represented by the set of points.

Hence, the searched curvature of the surface in a certain direction is the multiplicative inverse of the radius of the fitted osculating circle.

The osculating circle fitting is based on the geometric algebra in which a point is represented as a sphere with radius 0 and a plane is a sphere with infinite radius. These special characteristics of the geometric algebra permit to use the inner product of two geometric entities, like sphere and point, or plane and point, as a measure for the distance. For using this aspect, the osculating circle is handled like a sphere, which has the same radius and center like the circle.

This osculating circle fitting uses a least squares approach to minimize the sum of all inner products of the sphere and the points, to which the sphere is fitted. Additionally, since the representation of a plane and a sphere is equal, the fitting can also result in a plane instead of a sphere.

With this approach the symmetric matrix B can be constructed, where $\boldsymbol{p_i}$ denotes the ith point and m the number of points to which the sphere is fitted. For a more detailed deduction see [Hil06] and [Sei10].

$$B = \begin{pmatrix} \sum_{i=1}^{m} w_{i1} \cdot w_{i1} & \sum_{i=1}^{m} w_{i1} \cdot w_{i2} & \sum_{i=1}^{m} w_{i1} \cdot w_{i3} \\ \sum_{i=1}^{m} w_{i1} \cdot w_{i2} & \sum_{i=1}^{m} w_{i2} \cdot w_{i2} & \sum_{i=1}^{m} w_{i2} \cdot w_{i3} \\ \sum_{i=1}^{m} w_{i1} \cdot w_{i3} & \sum_{i=1}^{m} w_{i2} \cdot w_{i3} & \sum_{i=1}^{m} w_{i3} \cdot w_{i3} \end{pmatrix}$$

$$w_{ij} = \begin{cases} \boldsymbol{p_i} \cdot \boldsymbol{n} & j = 1 \\ -1 & j = 2 \\ \boldsymbol{p_i} \cdot \boldsymbol{x} - \frac{1}{2}\boldsymbol{p_i}^2 & j = 3 \end{cases}$$

The eigenvector of the smallest eigenvalue of this matrix B is the resulting vector $\boldsymbol{t}$ with which the radius r and the center $\boldsymbol{m}$ can be calculated:

$$\boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ t_3 \end{pmatrix} \qquad r = \frac{t_1}{t_3}$$

$$\boldsymbol{m} = \boldsymbol{x} + r \cdot \boldsymbol{n}$$

Usually a radius is a positive value, but in this case, r can be negative or positive. This is absolutely desired, since a curvature can be positive or negative. A negative curvature means that the curvature occurs in direction of the surface normal, a positive curvature means that the curvature occurs in the opposite direction. I.e. a point on a convex surface has negative curvatures in all directions and a point on a concave surface has only positive curvatures.

The principal curvatures of a plane surface cannot be correctly estimated with a sphere. Since the approach is based on geometric algebra, where a plane is a sphere with infinite radius, this case can be also be correctly identified. If a plane is fitted instead of a sphere t_3 is 0. Furthermore, an additional information is gained by t_2: if t_2 is 0, the fitted sphere or plane intersects the origin.

As mentioned before, the curvature of the curve, which was intended to be estimated, is the multiplicative inverse of the radius r. Hence, the calculation of the center $\boldsymbol{m}$ is actually not required for the algorithm.

4. NEW OSCULATING CIRCLE FITTING

The newly developed osculating circle fitting is also based on the geometric algebra and starts with the same matrix B. The further steps however are very different. We avoid the costly calculation of the eigenvalues of the matrix, through defining a restriction on the vector $\boldsymbol{t}$: we set $t_3 = 1$. With this restriction the resulting radius can still have any value. The only possible result, which cannot be obtained anymore is a plane, which has $t_3 = 0$. Nevertheless, we will see in a later step, that we can recognize a plane in another equation.

With the changed $\boldsymbol{t}$ we can construct a system of equations without calculating the eigenvalues. The values of the symmetric matrix B are denoted with the variables a up to f:

$$B = \begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix}$$

The system of equations is constructed as follows:

$$\begin{pmatrix} a & b & c \\ b & d & e \\ c & e & f \end{pmatrix} \cdot \begin{pmatrix} t_1 \\ t_2 \\ 1 \end{pmatrix} = 0$$

$$\Rightarrow \quad \begin{aligned} &I \quad && at_1 + bt_2 = -c \\ &II \quad && bt_1 + dt_2 = -e \\ &III \quad && ct_1 + et_2 = -f \end{aligned}$$

If we rearrange the equations we get:

$$\begin{aligned} &I \quad && at_1 + bt_2 = -c \\ &II \quad && 0 + t_2 = \frac{-e + \frac{bc}{a}}{d - \frac{b^2}{a}} \\ &III \quad && 0 + t_2 = \frac{-f + \frac{c^2}{a}}{e - \frac{bc}{a}} \end{aligned}$$

Since the system of equations is overdetermined, we can calculate t_2 in two ways. If we calculate t_1 for both possibilities by inserting t_2 into the first equation we obtain the following equations for t_1:

$$t_1 = \frac{be - cd}{ad - b^2} \quad and \quad t_1 = \frac{bf - ce}{ae - bc}$$

Due to the chosen value of t_3 the radius r actually has the same value as t_1.

$$t_3 = 1 \; and \; r = \frac{t_1}{t_3} \qquad \Rightarrow \qquad r = t_1$$

The radius of the osculating circle therefore has two possible results:

$$r = \frac{be - cd}{ad - b^2} \quad or \quad r = \frac{bf - ce}{ae - bc}$$

Even if the chosen t_3 apparently excluded the possibility of obtaining a plane we can recognize a plane in both equations. In both equations the denominator equals 0 if the fitting would result in a plane.

The both equations mostly result in the same value for r. The values only differ due to numerical instability of floating point numbers. Tests have shown that the second equation has less faulty fittings than the first equation. If we use the second equation to calculate the radius, the fitting needs in most cases less than 50 % of the time of the previous fitting method, but the accuracy of the fittings is reduced by an average of 2 %.

Another solution with impressive results, because it has not the same numerical instability, is to choose the equation with a better fitted osculating circle in every fitting, if the two equations for r do not have the same result.

We can determine a measure for the quality of the fitting, denoted as q, by calculating the sum of the distances d_i to all points to which the circle was fitted:

$$\boldsymbol{m} = \begin{pmatrix} m_x \\ m_y \\ m_z \end{pmatrix} = \boldsymbol{x} + r \cdot \boldsymbol{n} \qquad \boldsymbol{p_i} = \begin{pmatrix} p_{i_x} \\ p_{i_y} \\ p_{i_z} \end{pmatrix}$$

$$d_i = \left| \sqrt{\left(p_{i_x} - m_x\right)^2 + \left(p_{i_y} - m_y\right)^2 + \left(p_{i_z} - m_z\right)^2} - |r| \right|$$

$$q = \sum_{i=1}^{m} d_i$$

If we calculate q for both values of r, we can decide which has a better osculating circle and choose this r.

An argument against would of course be that additional time is needed for checking which osculating circle is better. Nevertheless, the improved results justify the raised effort. With this method the fitting still needs less than 75 % of the time of the previous fitting method and achieves results that are highly more accurate than the results of the previous fitting method. As we will see in the tests the results are about 35 % up to 300 % more accurate.

5. DYNAMIC VARIABLE ADJUSTMENT METHOD

The method is needed because the results of the osculating circle fittings are only useful if the set of points which the fittings had as input, were chosen well. This method tries to optimize those set of points by adjusting two variables of the algorithm: the surroundings-threshold and the plane-distance-threshold.

As described in the algorithm, the surroundings-threshold gives the limit, how small the distance of a point to $\boldsymbol{x}$ has to be, for including this point in the choice of the set of points.

The plane-distance-threshold gives a limit how small the distance of a point to a plane of one direction has to be, for including this point in the set of points representing the intersection of the corresponding plane and the surface.

This method tries to adjust the surroundings-threshold to achieve a well-proven number of points in the surroundings-set and simultaneously adjust the plane-distance-threshold to this number.

Various tests have shown that optimal results are achieved with around 375 points included in the surroundings and a plane-distance-threshold of 0,1. Therefore, we have chosen this combination 375/0,1 as the initial target combination of the dynamic variable adjustment method.

The method consists of 3 parts: The first part adjusts the surroundings-threshold for the subsequent point. The second part adjusts the target combination to more dense point clouds. The third part adjusts the plane-distance-threshold to the number of points in the surroundings-set.

We assume that the principal curvatures of all points of a point cloud are estimated sequently with the algorithm for principal curvature estimation. Therefore, we can use the fact that the denseness of the surrounding part of the point cloud does not differ much from one point to the next point.

This method is always executed right after the choice of the surroundings-set, hence before the planes in 16 directions are defined and before the set of points for the fittings are constructed. Therefore, the adjustment of the plane-distance-threshold can still affect the choice of the set of points, but the adjustment of the surroundings-threshold only affects the principal curvature estimation of the subsequent point.

In the following, the targeted number of points included in the surroundings-set is denoted as a_{target} and the targeted plane-distance-threshold as p_{target}. The actual detected number of points in the surroundings-set is denoted as a. The actual plane-distance-threshold is denoted as p and the surroundings-threshold as s.

Initially, a_{target} has the value 375, p_{target} the value 0,1 and s the value 4.

Adjustment of the surroundings-threshold

The surroundings-threshold has to be adjusted, because the number of points included in the surroundings should equal or at least almost equal the targeted number of points. If we assume that the points are more or less equally distributed in the point cloud we can calculate the optimal new surroundings-threshold with the help of the old surrounding-threshold and the number of points in the surroundings-set. The developed formula for the new surroundings-threshold is as follows:

$$s_{new} = \sqrt{\frac{{s_{old}}^2}{\frac{a}{a_{target}}}}$$

The best result would be achieved if the choice of the surroundings would be repeated after the adjustment. Unfortunately this step is very costly; therefore, the new surroundings-threshold is only used for the subsequent point. An exception is only the very first point for which the principal curvatures are estimated, since this is the only case where the point has no previous adjustment. For this reason, at the very first point, the choice of the surroundings-set is repeated after the adjustment.

Adjustment of the Target Combination

Tests have shown that for highly dense point clouds the initial chosen target combination 375/0,1 results in too small surroundings-threshold. Therefore, this combination is also adapted in some cases.

If the surroundings-threshold is adapted to a value lower than 2, the target combination is set to 750/0,05. If the surroundings-threshold is further adapted to a value lower than 1, the target combination is set to 1500/0,025.

However, the target combination should also be adapted to the original value if a less dense part of

the point cloud is reached. Therefore, the target combination is set back to 750/0,05 from 1500/0,025 or back to 375/0,1 from 750/0,05 if the surroundings-threshold is adapted to a value higher than 4.

Adjustment of the plane-distance-threshold

The plane-distance-threshold is always adjusted to an appropriate value, depending on the number of points in the surroundings-set and the targeted values:

$$p_{new} = p_{target} \cdot \frac{a_{target}}{a}$$

6. TESTS AND RESULTS

For evaluating the newly developed methods we have compared results of the new methods to results of the previous algorithm of [Sei10].

We have chosen 12 point clouds for the tests. The point clouds differ in the quantity of points and denseness of the point cloud. The surfaces represented by the different point clouds include various forms of curvatures. Therefore we achieve representative results, by testing with all 12 point clouds.

The principal curvatures of all points of the point clouds are known, since for evaluating the results we have to calculate the average difference to the correct values for all points of a point cloud.

Table 1 shows results without the dynamic variable adjustment method. The tests were made for several different combinations of the surroundings-threshold and the plane-distance-threshold. For each point cloud the table shows the best results of the best combination. Therefore, the table does not show that most of the other combinations resulted in average k_1/ k_2 differences of more than 0,01. This was the reason, why the dynamic variable adjustment method was developed.

Compared methods:

New01 is the algorithm with the new osculating circle fitting only using the second equation for r. **New02** is the algorithm with the new osculating circle fitting with selection of the better equation. **Previous** is the unchanged previous algorithm.

s/p is the chosen combination of the surroundings-threshold and plane-distance-threshold.

		Previous	**New01**	**New02**
1	s/p	6,0/0,3	6,0/0,3	6,0/0,5
	$Average\ \Delta k_1$	0,0050	0,0051	0,0031
	$Average\ \Delta k_2$	0,0050	0,0051	0,0023
2	s/p	6,0/0,3	6,0/0,3	6,0/0,5
	$Average\ \Delta k_1$	0,0032	0,0033	0,0022
	$Average\ \Delta k_2$	0,0023	0,0024	0,0014
3	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0016	0,0017	0,0011
	$Average\ \Delta k_2$	0,0007	0,0008	0,0006
4	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0018	0,0018	0,0015
	$Average\ \Delta k_2$	0,0007	0,0007	0,0006
5	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0016	0,0016	0,0015
	$Average\ \Delta k_2$	0,0011	0,0013	0,0011
6	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0007	0,0008	0,0006
	$Average\ \Delta k_2$	0,0014	0,0014	0,0013
7	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0017	0,0018	0,0017
	$Average\ \Delta k_2$	0,0008	0,0009	0,0008
8	s/p	6,0/0,1	6,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0014	0,0014	0,0005
	$Average\ \Delta k_2$	0,0023	0,0024	0,0010
9	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0025	0,0026	0,0010
	$Average\ \Delta k_2$	0,0025	0,0025	0,0017
10	s/p	4,0/0,1	4,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0017	0,0017	0,0011
	$Average\ \Delta k_2$	0,0032	0,0032	0,0015
11	s/p	4,0/0,1	4,0/0,1	4,0/0,5
	$Average\ \Delta k_1$	0,0051	0,0051	0,0050
	$Average\ \Delta k_2$	0,0014	0,0014	0,0013
12	s/p	6,0/0,1	6,0/0,1	4,0/0,1
	$Average\ \Delta k_1$	0,0011	0,0012	0,0003
	$Average\ \Delta k_2$	0,0017	0,0017	0,0005

Table 1. Test results with several possible combinations for s and p

As we can see, for most of the point clouds the **New02** method could achieve highly improved results. As we can see, for some point clouds no good results could be achieved with the limited number of s/p combinations.

The time needed for the fittings could also be improved. On average **New01** needed 50% and **New02** 75% of the time of **Previous**.

Table 2 shows the results with the dynamic variable adjustment method. In this table we can see that the dynamic variable adjustment method achieves good results and also that the new osculating circle fitting improved the principal curvature estimations.

	$Average\ \Delta k_1$		$Average\ \Delta k_2$	
	Previous	New02	Previous	New02
1	0,0061	0,0021	0,0041	0,0014
2	0,0037	0,0013	0,0028	0,0009
3	0,0010	0,0007	0,0008	0,0005
4	0,0008	0,0007	0,0006	0,0004
5	0,0012	0,0012	0,0006	0,0006
6	0,0018	0,0006	0,0021	0,0006
7	0,0006	0,0006	0,0003	0,0003
8	0,0010	0,0003	0,0011	0,0003
9	0,0004	0,0001	0,0005	0,0001
10	0,0012	0,0001	0,0014	0,0002
11	0,0007	0,0005	0,0002	0,0001
12	0,0014	0,0007	0,0015	0,0005

Table 2. Test results with the dynamic variable adjustment method

7. CONCLUSION AND OUTLOOK

In this paper we presented our new osculating circle fitting and the dynamic variable adjustment method. The goal of both developments was to improve the presented algorithm for principal curvature estimation.

The evaluation of the tests confirms that we have reached this goal. The new osculating circle fitting improves the results of the algorithm and is also faster than the previous one. The additional dynamic variable adjustment method improves the utilization of the algorithm for a whole point cloud. This method has also proven his worth.

The presented improvements were concentrated mainly on the raising of the accuracy. The algorithm could be further improved by expanding it with a Moving Least Squares approach like in [Ada03] and [Gue08]. Other improvements shall also highly raise the speed. For this task an implementation in CUDA or OpenCL are planned.

8. REFERENCES

[Ada03] Adamson, Anders and Alexa, Marc: Approximating and Intersecting Surfaces from Points. Eurographics Symposium on Geometry Processing (SGP), pages 230–239, 2003.

[Goi06] Gois, João Paulo, Tejada, Eduardo, Etiene, Tiago, Nonato, Luis Gustavo, Castelo, Antonio and Ertl, Thomas: Curvature-driven modeling and rendering of point-based surfaces. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI), pages 27-36, 2006.

[Gue08] Guennebaud, Gaël, Germann, Marcel and Gross, Markus: Dynamic Sampling and Rendering of Algebraic Point Set Surfaces. In Eurographics, pages 653–662, 2008.

[Hil06] Hildenbrand, Dietmar: Geometric Computing in Computer Graphics and Robotics using Conformal Geometric Algebra. Diss. Darmstadt 2006.

[Hor06] Hornung, Alexander and Kobbelt, Leif: Robust Reconstruction of Watertight 3D Models from Non-uniformly Sampled Point Clouds without Normal Information. Eurographics Symposium on Geometry Processing (SGP), pages 41–50, 2006.

[IEEE01] Institute of Electrical and Electronics Engineers**:** Std. 1241-2000 IEEE standard for terminology and test methods for analog-to-digital converters. chapter 3, pages 26-27, 2001.

[Kal07] Kalogerakis, Evangelos, Simari, Patricio, Nowrouzezahrai, Dere and Singh, Karan: Robust statistical estimation of curvature on discretized surfaces. Eurographics Symposium on Geometry Processing (SGP), pages 13–22, 2007.

[Kol04] Kolluri, Ravikrishna, Shewchuk, Jonathan Richard and O'Brien and James F.: Spectral Surface Reconstruction from Noisy Point Clouds. Eurographics Symposium on Geometry Processing (SGP), pages 11–22, 2004.

[Med05] Mederos, Boris, Amenta, Nina, Velho, Luiz and de Figueiredo, Luiz Henrique: Surface Reconstruction for Noisy Point Clouds. Eurographics Symposium on Geometry Processing (SGP), pages 53–62, 2005.

[Per09] Perwass, Christian**:** Geometric Algebra with Applications in Engineering. Berlin: Springer, 2009.

[Sei10] Seibert, Helmut, Hildenbrand, Dietmar, Becker, Meike and Kuijper, Arjan: Estimation of Curvatures in point sets based on geometric algebra. Angers: VISIGRAPP, International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications, 2010.

[Yan06] Yang, Yong-Liang, Lai, Yu-Kan, Hu, Shi-Min and Pottmann, Helmut: Robust Principal Curvatures on Multiple Scales. Eurographics Symposium on Geometry Processing (SGP), pages 223–226, 2006.

Investigating measures for transfer function generation for visualization of MET biomedical data

L. Svensson, I. Nyström, S. Svensson and I.-M. Sintorn
Centre for Image Analysis, Uppsala University and Swedish University of Agricultural Sciences
Box 337
SE-751 05 Uppsala, Sweden
E-mail: {lennart,ingela,stina,ida.sintorn}@cb.uu.se

ABSTRACT

In this paper, the question of automatically setting transfer functions for volume images is further explored. More specifically, the focus is automatic visualization of Molecular Electron Tomography (MET) volume images using one-dimensional transfer functions. We investigate how well a few general measures based on density, gradient, curvature and connected component information are suited for generating these transfer functions. To assess this, an expert has set suitable transfer function levels manually and we have studied how these levels relate to different characteristics of the selected measures for 29 data sets. We have found that the measures can be used to automatically generate a transfer function used to visualize MET data, to give the user an approximate view of the components in the image.

Keywords

Volume visualization, direct volume rendering, transfer functions, automatic visualization, molecular electron tomography

1 INTRODUCTION

Automatic visualization provides the means for screening large amounts of data in a short time by aiding the user in setting visualization parameters. Here, the goal is to investigate measures for automatically creating one-dimensional transfer functions that give good first renderings of Molecular Electron Tomography (MET) data. These should highlight the most important information, i.e., the molecular surface of proteins, and still show other variations in the imaged sample. The visualization should be a starting point for interactive adjustments. Primarily, the focus is to identify measures which generate an appropriate opacity function.

MET allows for studying the structure and flexibility of molecules and macromolecules in solution (in vitro) as well as in tissue samples (in situ). The imaging technique reveals material density with a resolution as low as a few nanometers. For determining how the proteins function in their natural environment, tissue samples are analyzed directly using MET.

For determining molecule flexibility and dynamics, molecules in solution are analysed, see Klaile [5] for a recent example. In order to investigate, explore, and analyse the complex data, adequate visualization of the data is required.

For general automatic transfer function generation, the "transfer function bake-off" [7] presents four approaches: (1) trial-and-error, (2) data-centric without model, (3) data-centric with model, and (4) image-based. Multi-dimensional transfer functions based on curvature have been introduced by Kindlmann [4], transfer functions specified as the sum of Gaussians were presented by Kniss [6], schemes based on topology differentiation have been suggested [11, 12], Rezk Salama presented an approach [8] which focuses on parameters relating to the user's domain knowledge. The mentioned methods move transfer function generation close to identification and segmentation problems.

To our knowledge, no transfer function generator tailored for MET volumes has so far been suggested. The volumes are usually rendered with direct volume rendering with a 1-D transfer function that is manually set. Often, the pre-integration step, that was presented by Engel [2], is left out, leading to notable visualization artefacts. The transfer function generation problem is highly relevant for this type of data since the volumes are difficult to interpret, see Figure 1, due to low contrast, small objects, missing data, etc. Another problem with this type of data is that many factors af-

fect the density values in the volumes. Among the influencing factors are the energy of the electrons hitting the sample, the sample thickness and composition, the detector, and the MET reconstruction algorithm.

We address this issue by following the second approach in the bake-off paper [7], the "data centric without model" approach. The idea is to make approximate distinctions between objects using measures in the density range of the volume image. The aim is to establish relations for generating a transfer function in this domain automatically. We have chosen a one-dimensional transfer function because of the intrinsic property that densities are remapped to opacities in a consistent way. Multi-dimensional transfer functions can be useful to differentiate between regions, but there is a less clear connection to the underlying density. If a model-based approach was employed, where individual components would be identified using various means of thorough image analysis, it would have enabled more elaborate fine-tuning of the visualization. This would however be at the expense of having a more complex solution and time-consuming algorithm, as well as possibly lower generality.

Using measures in the density range is similar to the approach presented by Bajaj [1], but the calculated functions are different and are also suggested to be used in a different way. We focus on automatic extraction of isovalues, whereas Bajaj suggested his measures to be used for interactive isovalue selection and volume data exploration. We have studied four different measures based on density, gradient, curvature and connected component functions. The interesting point is how features of these measures relate to manually chosen levels by an expert. For the density histogram, it has been investigated what percentiles the manually set levels correspond to, whereas for the gradient, curvature and connected component measures, different features of the functions have been correlated to the manually set level. We have used 29 volumes in the tests.

2 IMAGE DATA

In MET, an electron microscope is used to capture 2D micrograph images from different angles of a sample. This results in a so called tilt series. The sample is a very thin frozen or chemically embedded slice. A back projection technique is applied to reconstruct a 3D image of the sample, which is then refined in an optimization procedure resulting in a *MET volume* [10]. Its scalar values correspond to the density of the sample. The complete process from sample preparation to a final volume is a long and tedious process, so these volumes are not available in large quantities.

MET volumes are difficult to interpret for a number of reasons: the resolution is relatively low — each protein is represented by a small number of voxels; the contrast is low as electron irradiation destroys the sample, which means that the total dose used to acquire the micrographs must be kept low; the MET volume suffers from missing data artefacts as the electron microscope limits the angular range to 120° – 140°; and the density levels in the MET volume are relative and not absolute. For an untrained eye, the volumes often seem to only contain a large number of blobs of varying size and shape.

Data set	No of invest. volumes	No of molecule instances per vol.
IgG	3	$\sim$2
RNAP II	3	$\sim$10
CEACAM1	6	$\sim$90
TMV	17	$\sim$2

Table 1: Investigated MET volumes

Another challenge when visualizing MET volumes compared to, e.g., MRI volumes, is that the molecules cannot be studied individually using visible light. This means there is no ground truth to refer to regarding how they should be visualized.

We investigate and evaluate the chosen measures using 29 MET volumes from four different studies of proteins in solution. All MET volumes were reconstructed using the constrained maximum entropy tomography method [10], giving a few nanometer resolution. The proteins are described in brief below. Since this kind of data is difficult and labour intense to generate, 29 volumes should be considered as a relatively large set of volumes. See Table 1 for a list of the number of respective protein images and molecules.

The RNA Polymerase II (RNAP II) is a large, fairly round macromolecule responsible for mRNA synthesis in eukaryotic cells. In the investigated MET volume, a RNAP II macromolecule has a $\sim$ 21 voxel diameter.

The Immunoglobulin G (IgG) antibody is a smaller macromolecule. Antibodies are crucial parts of our immunological defence system, e.g., IgG binds to foreign agents such as virus particles and targets them for destruction. An IgG antibody has three roundish parts of equal size connected at one center point. Two of the arms are fragment antigen binding arms and one is a fragment crystallisable stem. In the investigated MET volume, the smallest round subpart has a $\sim$ 10 voxel diameter. See Sandin [9] for details.

The carcinoembryonic antigen related cell adhesion molecule 1 (CEACAM1) is even smaller than IgG. It is a transmembrane receptor involved in binding with other cells. In the investigated MET volumes, CEACAM1 occur as monomers, one molecular unit, with a volume of $\sim$ 580 voxels, or dimer, two molec-

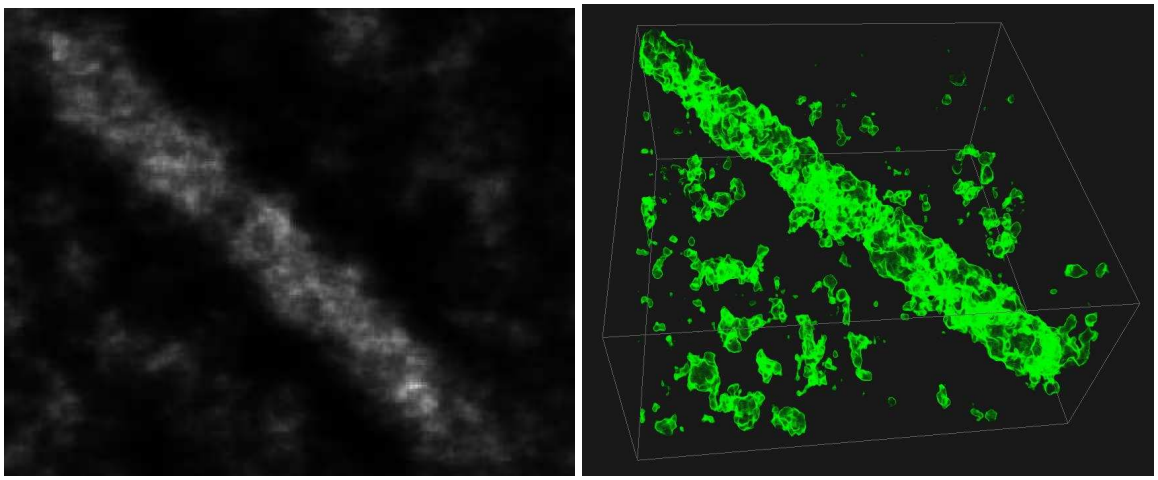


Figure 1: The Tobacco Mosaic Virus (TMV) reconstructed using Molecular Electron Tomography (MET). Left: a slice of a MET volume is shown. Right: a volume visualization with a manually set transfer function.

ular units linked together, with a volume of ~ 1160 voxels. See Klaile [5] for details.

The Tobacco Mosaic Virus (TMV) is a larger, tubular structure. In the investigated MET volume, the TMV has ~ 32 voxel diameter. It infects, e.g., tobacco plants.

The MET volumes in the study are approximately 256^3 voxels. The protein concentrations are such that we expect ten RNAP II molecules, one or two IgG antibodies, and 80–100 CEACAM1 molecules in each MET volume. For TMV, one to three structures are present in each volume.

3 INVESTIGATION

A MET volume can be described as $f : \mathbb{N}^3 \rightarrow \mathbb{R}$, not considering the limitations of digital number representation, while the used transfer function corresponds to $g : \mathbb{R} \rightarrow \mathbb{R}^4$. The four output components of the transfer function are the three color channels, RGB, and the opacity.

We suggest to render a MET volume with a transfer function built using two primitives, a Gaussian and a piece-wise linear function going from transparent to opaque, see Figure 2 (top). The idea behind the Gaussian is that it should correspond to the most central surface level for the molecules of interest. This we denote the *primary level*. With a high concentrated peak, the Gaussian will have a similar effect as an iso-surface, but both reveal the shape of objects and give an idea about the blurriness around it, i.e., if the surface is part of a sharp change or a smooth transition. The piece-wise linear function has two roles, partly to reveal more of the MET volume and partly to highlight outliers. To visualize this, a linear ramp with low opacity slope is suggested, starting at the *secondary level*, to not disturb the surface visualization, while a linear ramp with high slope in a different color is recommended for the outlier density range, starting at the *outlier level*.

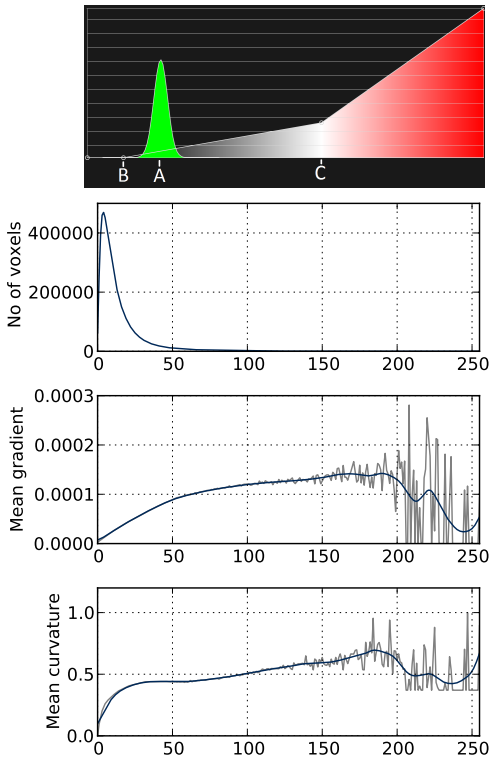


Figure 2: Example of transfer function (top). Plots of the histogram and the gradient and curvature functions of a MET volume containing a Tobacco Mosaic Virus (bottom).

To set the positions for these primitives, the following estimates need to be extracted from the chosen measures:

- Primary level (center of the Gaussian, 'A' in Figure 2)
- Primary level standard deviation (width of the Gaussian)
- Secondary level start (start of low slope ramp, 'B' in Figure 2)
- Outlier level start (end of low slope ramp, start of high slope ramp in different color, 'C' in Figure 2)

We have focused on localizing the primary level. For the secondary level, the outlier level and the primary level standard deviation, we have not yet revealed any correlation. Therefore, we suggest to use a fixed ratio to the primary level for these levels. We suggest to set the secondary level at half the primary level, the outlier level at double the primary level, and the primary standard deviation to $\pm 10\%$ around the primary level.

To establish a relationship between that density level and the measures, the primary level was manually set by an expert to give a good visualization of the MET volumes. We have investigated direct correlation between the manual levels and the measures, i.e., performed analysis of the different measures separately.

In Figure 2, the density histogram, the mean gradient and the mean curvature are shown for a TMV volume. The number of bins for calculating the different functions is set to 256, which is higher than the accuracy of the extracted estimates.

A central point to investigate in these plots is whether they are based on multiple distributions, which could arise from differences between spurious blobs in the volumes and shapes of actual molecules. In the following subsections the measures are described. The results are shown and discussed in Section 4.

The density histogram is the basis for measure 1. This is a "standard" histogram that shows the number of voxels that fall within a certain density bin. We investigate how the expert levels are distributed in the histogram. The ideal, but a little boring, result would be that the levels correspond to a single histogram percentile. This would mean that it would suffice to visualize a fixed fraction of a MET volume.

The number of components with size filtering is the basis for measure 2. This creates a function over the density values as for the histogram, representing the number of components at one density bin within the specified size range. To obtain this number, the lower bin value is used as threshold and all voxels with a value equal to or higher than this should belong to a component. If two voxels are 26-connected, then they belong to the same component. The size filter is applied to increase the "hit rate" in the density region of most interest. To extract the primary level estimate from this function, the position of the maximum is simply used. The size estimates have been manually calculated from the object diameters and their basic shape (round or tubular), with some margin.

The mean gradient histogram is the basis for measure 3. A gradient magnitude image is calculated using the first order derivative of a Gaussian kernel with a sigma related to the approximate diameter in voxels of the components of interest in the images. The gradient magnitude for each voxel is

$$|\frac{\partial f}{\partial x}| + |\frac{\partial f}{\partial y}| + |\frac{\partial f}{\partial z}| \tag{1}$$

where f is the density. The mean gradient magnitude value for the set of voxels within the density bin is then calculated. From the plots of this function, we have identified a reoccurring plateau starting approximately around the primary level chosen by the expert. We have defined the starting point for this plateau as the first local minimum of a smoothed derivative of the mean gradient measure. The smoothing has been done with a Gaussian filter with sigma set to 5, half the size of one subpart of the IgG molecule.

The mean curvature histogram is the basis for measure 4. It follows the curvature approach used by Kindlmann [4]. Essentially, these values measure the mean curvature for voxels at a particular intensity level. A value of zero would mean that the voxels within the corresponding intensity interval is not a part of an isosurface that has a strong curvature.

The first and second order partial derivatives needed for the gradient $\mathbf{g}$ and the Hessian matrix $\mathbf{H}$ are calculated using combinations of first and second order derivatives of the Gaussian kernel. Then the surface normal

$$\mathbf{n} = \mathbf{g}/|\mathbf{g}| \tag{2}$$

and the projection matrix

$$\mathbf{P} = \mathbf{I} - \mathbf{n}\mathbf{n}^T \tag{3}$$

where $\mathbf{I}$ is the identity matrix, are calculated. The matrix $\mathbf{P}$ projects onto the tangent plane of the isosurface. Next, the matrix

$$\mathbf{G} = \text{-}\mathbf{PHP}/|\mathbf{g}| \tag{4}$$

referred to by Kindlmann as the geometry tensor, is formed and from that the trace T and Frobenius norm F. The mean curvature is calculated as:

$$\kappa_1 = (T + \sqrt{F^2 - T^2})/2 \tag{5}$$

$$\kappa_2 = (T - \sqrt{2F^2 - T^2})/2 \tag{6}$$

Then, the mean curvature for a single voxel is $(\kappa_1 + \kappa_2)/2$. As in the mean gradient calculation, the total mean curvature for one density bin is calculated to acquire the measure. To match it with the primary level,

Dataset	Sigma	Size threshold (voxels)
RNAP II	2.0	2500
IgG	1.0	500
CEACAM1	1.0	500
TMV	2.0	50000

Table 2: Parameters for Gaussian and size filtering

we have identified a plateau for this measure as well. We have defined its starting point in the same way as for the gradient measure, i.e., as the first local minimum of a smoothed derivative of the mean curvature measure.

A priori information has been used when calculating the gradient and curvature measures, in the form of setting sigma to 1/10 of an estimation of the smallest component diameter. This will preserve the main structure of the components, but remove some of the noise.

For volume rendering, pre-integrated ray casting was used. Ray casting provides the possibility to visualize more information than an isosurface rendering. We claim that the user can get a better feeling for the data in the volume, by for example also taking densities around an isolevel into consideration. In order to still have an exact visualization with transfer functions which can contain high frequency changes, the pre-integration step is necessary. The methods were implemented in C++ partly using routines from the National Library of Medicine Insight Segmentation and Registration Toolkit (ITK) [3].

4 RESULTS AND DISCUSSION

We have not found any clearly multimodal distribution in any of the feature functions. From visual inspection of the density histogram, the distribution is close to a gamma distribution regardless of what kind of molecules there are in the solution. For the other three feature measures, there is more variation. In Figure 9, the correlation between the expert levels and the measures are shown. The used manually set filter parameters are given in Table 2. For the immediate visualization of a volume when it is opened, it should not be required to enter such information, but preliminary tests show that using a standard value for sigma will still give feasible results.

Measure 1 was visually evaluated, whereas for measures 2-4, a performance index has also been calculated for each of the measures. First a line $y = kx + m$ has been fitted to the data using a least square error norm. The performance index is calculated as

$$P = 100\frac{k}{\text{error}} \qquad (7)$$

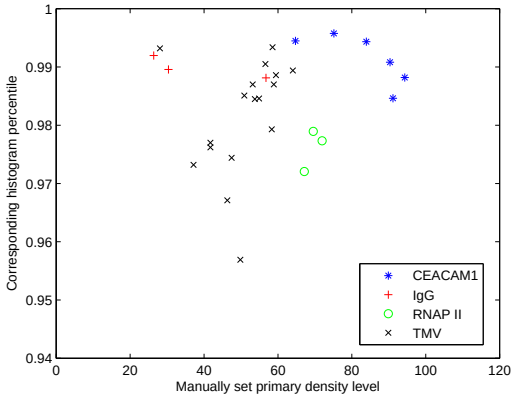


Figure 3: Primary levels chosen by an expert for the 29 investigated volumes plotted against their corresponding density histogram percentiles.

This index evaluates the discriminative power of a measure, but it is nothing more than an inverted correlation to the error, which is multiplied with k to remove the effect of the value range of the measure. The error is calculated as the mean of absolute errors.

1. Density histogram, Figure 3: The aim is to find a decorrelation when calculating the percentile value from the manual level. That is, the manual levels would optimally correspond to a single percentile level, forming a horizontal line. In Figure 3, a decorrelation tendency can be seen as not all the data seem to be spread around any diagonal "correlation line". Another observation is that the primary level is always above 95% for the investigated data sets. A comparison of an expert visualization and a visualization using the 99-percentile of the density histogram as the primary level is shown for RNA Polymerase II in Figures 4 and 5, respectively. The same comparison is shown for TMV in Figures 6 and 7, respectively.

2. Connected components, Figure 9: Considering all data sets, there is only some correlation. When excluding the sets with the largest molecules, the TMV data sets and the RNA Polymerase II data sets, a linear tendency can be seen. The performance index is 4.1 using all data sets.

3. Gradient, Figure 9: This measure exhibits the highest correlation tendency to the expert primary level, with a performance index of 7.5. This indicates that the molecules of interest have a more homogeneous internal density structure than the lower intensity artefacts.

4. Curvature, Figure 9: For the curvature measure, there seems to be a weak linear tendency. The performance index is 3.0. Hence, the difference in isosurface curvature between the biological molecules and other structures does not seem very significant.

Combining the measures using their performance indices as averaging weights gives the result shown in Figure 8. The performance index of this combined measure is 7.4, which is lower than for the gradient based measure.

When testing on an Intel E5430 2.66GHz CPU, loading and processing a 256^3 volume took approximately one minute. In an interactive application, the generation of a transfer function needs to be faster, a few seconds would be preferable, which should be feasible with optimization.

5 CONCLUSION

Four measures have been investigated regarding their potential for automatically generating a first visualization of MET volume data. We see all as interesting measures in this context, but the gradient based measure stand out as giving the best estimate of the primary level. We therefore suggest to use gradient based analysis for best accuracy when setting the primary level. Another simple but interesting result is that the primary level of interest for the investigated data sets is always in the top 5% of the volumes, in terms of density. Since this percentile measure is fast to compute, it is a good basic measure for instant automatic visualization, especially of large MET volumes. It could also be used as a control measure when calculating the primary level in a more exact way.

It is suitable that the gradient measure with the highest performance index also is the second easiest measure to calculate, after the histogram percentile, although it still takes around 15 seconds for a typical volume to be processed for this measure. In terms of algorithmic complexity, the gradient measure is based on separable filtering, so it will scale nicely for larger volumes.

Our next step is to step up a scale in terms of feature calculation, to make the distinction of objects of interest and other structures easier. One path would be to use region growing methods and explore different components using suitable shape descriptors.

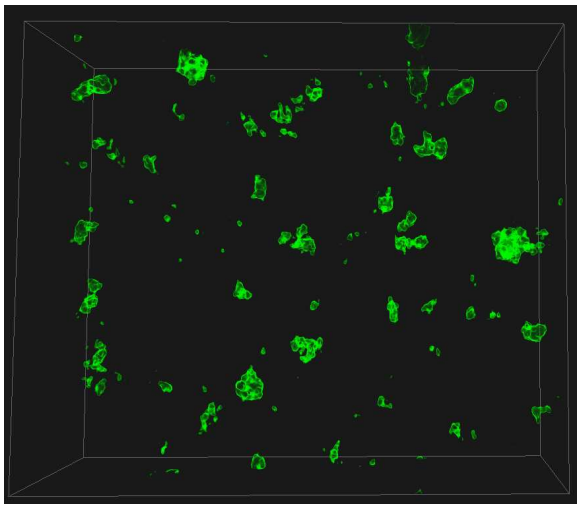


Figure 4: Visualization of a RNA Polymerase II volume, with an expert set primary level.

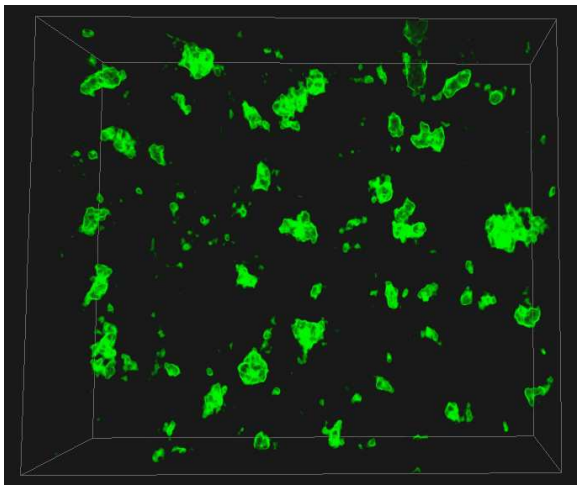


Figure 5: Visualization of the same RNA Polymerase II volume as in Figure 4, setting the primary level at the 99-percentile of the density histogram, but also showing $\pm$0.8% around that level.

ACKNOWLEDGEMENTS

This work is funded through the *Visualization Program* by Knowledge Foundation, Vårdal Foundation, Foundation for Strategic Research, VINNOVA, and Invest in Sweden Agency.

We would like to thank Roger Kornberg Laboratory, Dept. of Structural Biology at Stanford University Medical School, USA; Elenor Hauzenberger and Lars-Göran Öfverstedt at Sidec AB, Sweden; Sara Sandin, et al. [9]; Esther Klaile, et al [5], for providing the data and to Daniel Evestedt, SenseGraphics AB, for partly implementing the pre-integrated ray caster. We also thank the anonymous reviewers for their constructive criticism.

REFERENCES

[1] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In *Proc. Vis. '97*, pages 167–173, 1997.

[2] Klaus Engel, Martin Kraus, and Thomas Ertl. High-quality pre-integrated volume rendering using hardware-accelerated pixel shading. In *SIGGRAPH/Eurographics Workshop on Graphics Hardware*, pages 9–16. ACM, 2001.

[3] Luis Ibanez, William Schroeder, Lydia Ng, and Josh Cates. *The ITK Software Guide*. 2005.

[4] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Möller. Curvature-based transfer functions for direct volume rendering:

methods and applications. In *Proceedings Vis. '03*, page 67, 2003.

[5] Esther Klaile, Olga Vorontsova, Kristmundur Sigmundsson, Mario M. Müller, Bernhard B. Singer, Lars-Göran Öfverstedt, Stina Svensson, Ulf Skoglund, and Björn Öbrink. The CEACAM1 N-terminal Ig domain mediates cis- and trans-binding and is essential for allosteric rearrangements of CEACAM1 microclusters. *Journal of Cell Biology*, 187(4):553–567, 2009.

[6] Joe Kniss, Simon Premoze, Milan Ikits, Aaron Lefohn, Charles Hansen, and Emil Praun. Gaussian transfer functions for multi-field volume visualization. In *Proc. Vis. '03*, page 65, 2003.

[7] Hanspeter Pfister, Bill Lorensen, Chandrajit Bajaj, Gordon Kindlmann, Will Schroeder, Lisa Sobierajski Avila, Ken Martin, Raghu Machiraju, and Jinho Lee. The transfer function bake-off. *IEEE Comput. Graph. Appl.*, 21(3):16–22, 2001.

[8] Christof Rezk Salama, Maik Keller, and Peter Kohlmann. High-level user interfaces for transfer function design with semantics. *IEEE TVCG*, 12(5):1021–1028, 2006.

[9] Sara Sandin, Lars-Göran Öfverstedt, Ann-Charlotte Wikström, Örjan Wrange, and Ulf Skoglund. Structure and flexibility of individual immunoglobulin G molecules in solution. *Structure*, 12(3):409–415, 2004.

[10] Ulf Skoglund, Lars-Göran Öfverstedt, Roger M. Burnett, and Gérard Bricogne. Maximum-entropy three-dimensional reconstruction with deconvolution of the contrast transfer function: A test application with adenovirus. *Journal of Structural Biology*, 117:173–188, 1996.

[11] Gunther H. Weber, Scott E. Dillard, Hamish Carr, Valerio Pascucci, and Bernd Hamann. Topology-controlled volume rendering. *IEEE TVCG*, 13(2):330–341, 2007.

[12] Jianlong Zhou and Masahiro Takatsuka. Automatic transfer function generation using contour tree controlled residue flow model and color harmonics. *IEEE TVCG*, 15(6):1481–1488, 2009.

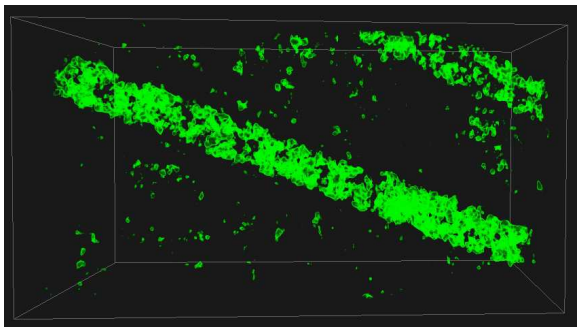


Figure 6: Visualization of a MET volume with a Tobacco Mosaic Virus (TMV) using an expert set primary level.

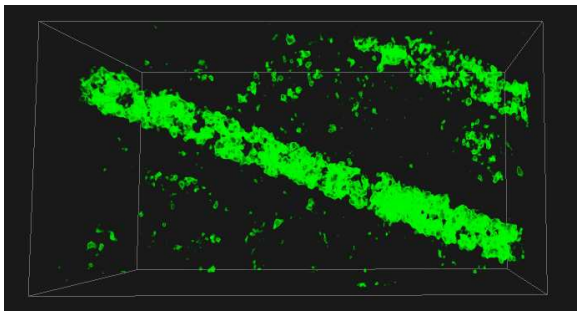


Figure 7: Visualization of the same MET volume of a TMV as in Figure 6 setting the primary level at the 99-percentile of the density histogram, but also showing $\pm$0.8% around that level.

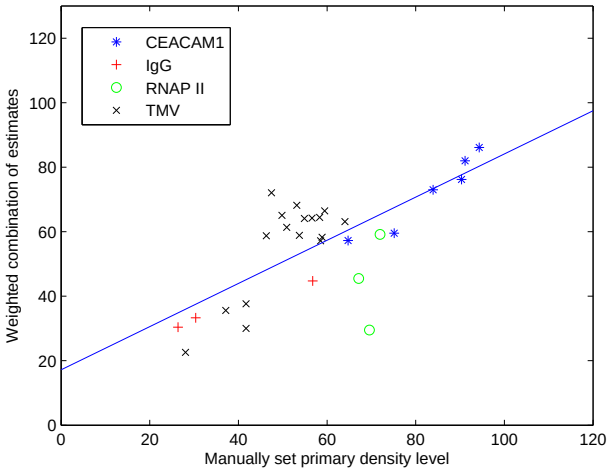


Figure 8: How a weighted combination of the measures correlates to the expert set primary level. The fitted line is used to calculate the performance index of the combined measure.

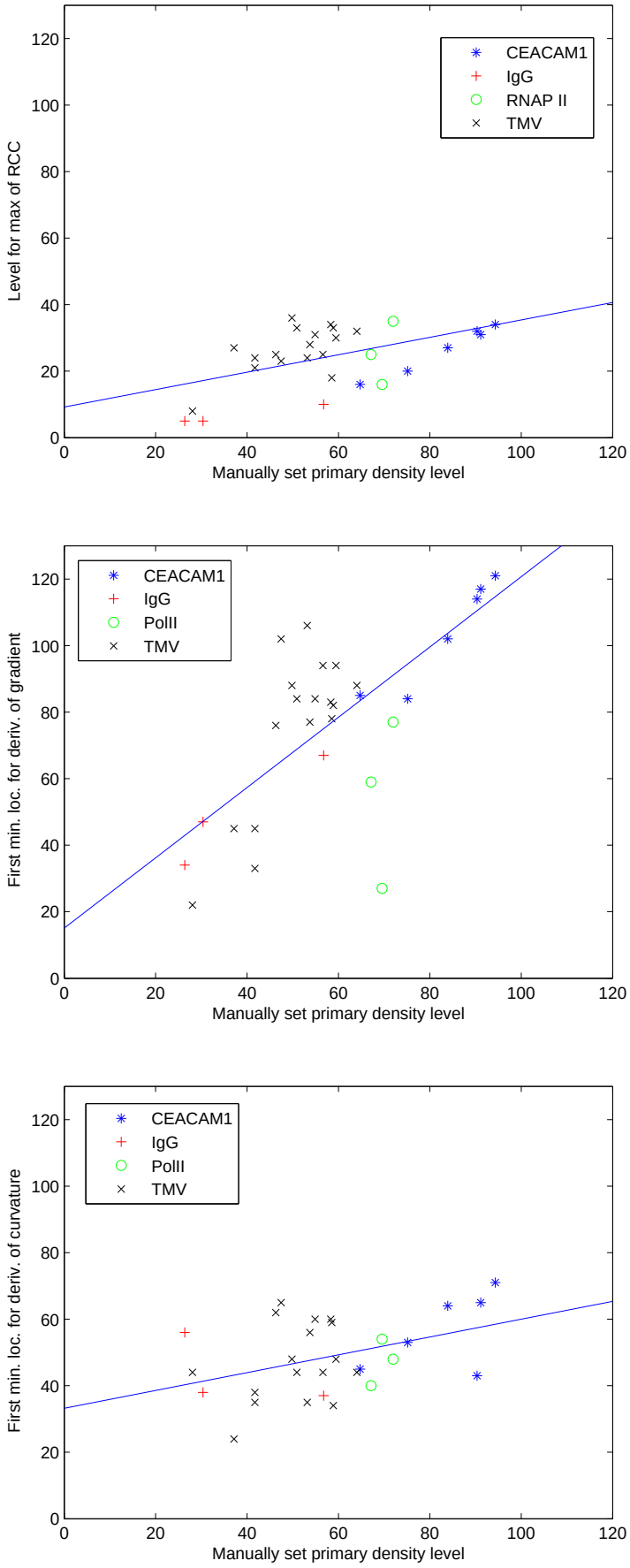


Figure 9: How the expert set primary level correlates to the extracted measures. The fitted line is used for calculating the performance index of each measure. Top: For the connected component measure there is a weak correlation to the expert level. Middle: The gradient measure shows correlation. Bottom: The curvature measure shows a weak correlation.

A Method for Storing Clustering Information of Model Simplification in GPUs

Takayuki Kanaya
Hiroshima International University[1)]
555-36, Kurose Gakuendai
Higashi Hiroshima, Hiroshima
Japan (739-2695)
t-kanaya@hw.hirokoku-u.ac.jp

Tomoaki Taniguchi
Osaka Institute of Technology[2)]
1-79-1, Kitayama
Hirakata, Osaka
Japan (573-0196)
taniguchi@ggl.is.oit.ac.jp

Yuji Teshima
Sasebo National College of Technology[3)]
1-1, Okishinchou
Sasebo, Nagasaki
Japan (857-1171)
teshima@post.cc.sasebo.ac.jp

Koji Nishio
Osaka Institute of Technology[2)]
nishio@is.oit.ac.jp

Kenichi Kobori
Osaka Institute of Technology[2)]
kobori@is.oit.ac.jp

ABSTRACT

A vertex-clustering simplification is a kind of model simplification. It is difficult for the vertex-clustering simplification to simplify complex models in real-time, although it is known as a very fast method. In addition, it is also difficult for the vertex-clustering simplification to control the number of faces. It synthesizes vertices in each cluster. Therefore, models sometimes consist of the unexpected number of faces. In recent years, Graphics Processing Units (GPUs) have grown so significantly in performance that both the computational speed and the computational accuracy improve spectacularly. GPUs have programmable units such as vertex shaders and geometry shaders. With shaders, GPUs can be used not only for graphics rendering but also for general purposes.

In this paper, we propose a real-time simplification algorithm for complex models of 3D objects by using a GPU whose performance gets better these days. First, vertex-clustering information is stored to video memory on a GPU. Next, the faces are reduced by the vertex-clustering information using a programmable shader, depending on the level of detail which a user defined. We also discuss a method to control the number of faces easily.

Keywords

Simplification, GPU, Vertex-Clustering, Real-Time rendering

1. INTRODUCTION

In recent years, it has been easy to express complex models of 3D objects in product design, simulation, medicine and games electronically, due to progressive computer technology and progressive computer graphics. However, it is still difficult to render them in real time. Therefore technology is required to change the level of detail (LOD) of multi-resolution representations of models according to a user's needs.

A vertex-clustering algorithm is known as a kind of model simplification. While the vertex-clustering algorithms feature low computational costs, they have 2 disadvantages. One is that model simplification has traditionally not been viewed as a real-time rendering on CPU, the other is that it is difficult for a user to control the degree of the LOD. The vertex-clustering algorithm has been proposed by Rossignac and Borrel [Rossignac and Borrel 1993]. This is the method where a cell, which includes all the vertices that exist in 3D space, is uniformly divided; all vertices within each grid cell are collapsed to a single representative vertex, which may be one of the input vertices or some weighted combination of the input vertices. Some triangles degenerate to edges or points. A number of other vertex clustering algorithms have also been proposed. Low and Tan have proposed a modified vertex-clustering algorithm using floating cells rather than a uniform grid [Low and Tan 1997]. The floating-cell clustering leads to more consistent simplification. Since the importance of vertices controls the positioning of clustering cells, the

unpredictable simplification artifacts are greatly reduced. Luebke and Erikson have proposed the algorithm using a hierarchical grid in the form of an octree, which is called Tight Octree [Luebke and Erikson 1997]. This vertex tree allows for a dynamic, view-dependent adaptation of the level of detail. Kanaya and Kobori have proposed the improved Tight Octree algorithm allowing for cell size [Kanaya and Kobori 2002]. This method is easier to control the LOD level. Lindstrom has used QEM(quadric error metrics) [Garland 1999] to improve the positioning of the representative vertices [Lindstrom 2000]. Shaffer and Garland have proposed BSP-Tree partitioning to control the LOD level [Shaffer and Garland 2001], and have also used QEM to improve the positioning of the representative vertices [Garland and Shaffer 2002]. Mesh simplification has been a slow, CPU-limited operation performed as a pre-process on static meshes.

In recent years, GPUs have grown so significantly in performance that the computational speed and the computational accuracy improve spectacularly. Additionally, a general–purpose GPU, such as numerical calculation, modeling, have been studied by a Programmable Shader's appearance.

[DeCoro and Tatarchuk 2007] is famous as a model simplification using GPU. This method is based on [Lindstrom 2000] adopted to the novel GPU pipeline. Additionally, they have proposed the algorithm for variable level-of-detail, called probabilistic octree. This method is much faster than CPU-based algorithm.

We present an algorithm overcoming the 2 disadvantages of vertex-clustering algorithms. For this, we present a novel general-purpose data structure designed for a GPU. At the same time, we present a method to linearize a relationship between a level which a user defined and the number of faces. As a result, our algorithm is faster and the number of faces is easier to control.

2. Our algorithm

We outline our algorithm, which consists of the following five steps:

1. The space division algorithm.
2. Synthesis of vertices in each cluster.
3. Updating level for controlling the number of faces.
4. Storing the clustering information in a GPU.
5. Generating the simplified models.

The above procedures (1) to (2) are performed on the CPU; the procedures (3) to (5) are performed on the GPU.

2.1 Space division algorithm

We applied the algorithm of [Kanaya and Kobori 2002], which we have proposed, to an arbitrary 3-D space division. The space division algorithm consists of the following five steps:

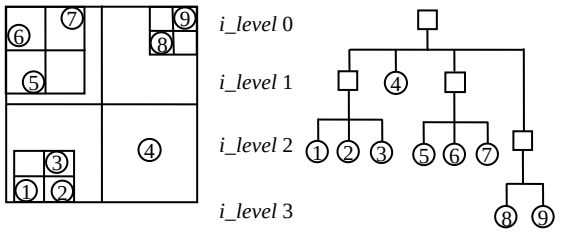
1. A cell, which includes all the vertices that exist in 3D space, is generated. The cell is a minimized cuboid, which is divided by domain parallel to the planes of the x-y coordinate, the y-z coordinate, and the z-x coordinate so that two or more vertices can be included.
2. The cell is equally divided by eight, and then eight cuboids are made.
3. The degree of the LOD (level of details) is determined by the longest edge of each cuboid. This is estimated by the following equation.

$$i\_level = \mathrm{INT}(\log_2 \frac{Length_{\max}}{Length_{current}}) \quad (1)$$

where, INT() is an integral function. The $Length_{max}$ is the longest length of cell that envelops all vertices. The $Length_{current}$ is the longest length of the considered cell. The *i_level* is integer value.

4. The number of vertices belonging to each cuboid is equally divided into eight; dividing will end if the number of the vertices, which exists in the cuboid, is only one.
5. Otherwise, each remaining cuboid is changed into a cell and processing returns to step 2.

By this procedure, a tree can be generated as shown in Figure 1. This tree is an octree, since a cell is always divided by eight. Considering the degree of the LOD in this octree, the root node is an *i_level* 0 and the degree of the LOD becomes larger as the node is closer to a leaf node. The octree is clustering information.



(a) The space division. (b) An octree of (a).

Figure 1. A generated octree using a spatial partitioning. (In 2D space)

2.2 Synthesis of vertices in cluster

A method for the synthesizing of vertices in cluster is based on [Kanaya and Kobori 2002]. This is why we simply want to compare CPU-based algorithms with GPU-based algorithms. It is possible to use [Lindstrom 2000] instead of Kanaya and Kobori for accuracy improvement.

2.3 Updating level for controlling the number of faces

To make it easier for the user to define the number of faces, we propose a way of linearizing a relationship between a level which a user defined and the number of faces as follow.

First, we cluster the internal nodes which *i_level* is same. However, the longest length of cell varies according to cell in spite of the same *i_level*. The internal nodes are arranged in descending order of the length in each cluster which has the same *i_level*. Then, we call the level which has the rank in descending order in each cluster "*r_level*". The *r_level* is a real number. The *r_level* of each internal node m "r_level_m" is calculated by equation (2).

$$r\_level_m = i\_level_{before} + \frac{1}{number_i} \times k \quad (2)$$

where, i_level_{before} is the *i_level* of each internal node. *k* is a rank of each internal node after arranging in descending order. $number_i$ is the number of internal node in arbitrary *i_leveli*. The leaf level is not updated.

2.4 Storing the clustering information in a GPU

The clustering information which was generated by the above process is stored in a texture on the GPU as texture. We prepare 2 kinds of textures for storing the clustering information. One texture is called "Tree-buffer"; the other texture is called "Leaf-buffer". The Tree-buffer stores the information of the internal nodes in an octree. The Leaf-buffer stores the information of the leaf nodes. We illustrate the way to store the information of nodes in textures in Figure 2.

First, we describe a method used for storing the internal nodes in the Tree-buffer. Each texel of the Tree-buffer stores the synthesized coordinate of vertices and the r_level_m as shown in Figure 3, while arbitrary internal nodes with parent-child relationship arrange a column parallel to the *v* axis, as shown in Figure 2.

Next, we describe a method used for storing the leaf nodes in the Leaf-buffer. Each texel of the Leaf-buffer stores a coordinate of vertices of leaf node in the octree and a pointer whose value refers directly to

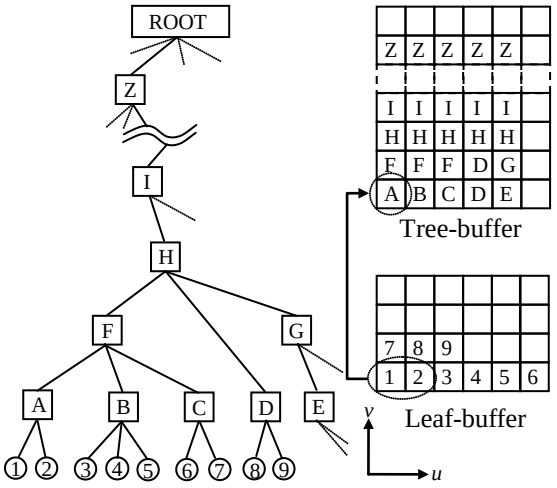


Figure 2. A stored state in the Tree-buffer and the Leaf-buffer.

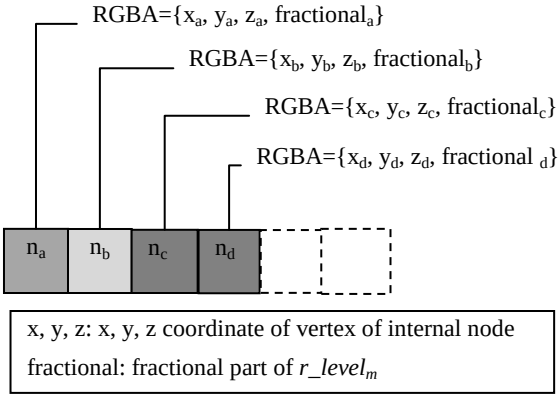


Figure 3. Method used for storing information in the Tree-buffer.

the parent node in the Tree-buffer as shown in Figure 4, in order from the bottom left as shown in Figure 2. For example, as we show in Figure 2, the leaf node 1, 2 hold a pointer whose value refers directly to the internal node A which is the parent for both nodes, respectively. As a result, it is easy to refer to a node

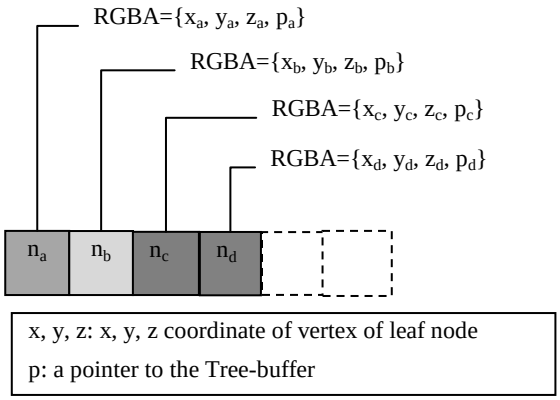


Figure 4. Method used for storing information in the Leaf-buffer.

of a level a user wants, when the texture coordinate of the terminal internal node is known.

2.5 Generating the simplified models

We describe the way for simplifying models by using the Tree-buffer and the Leaf-buffer on a GPU. First, we outline the generating procedure of the simplified model.

1. Calculate "*r_level*" of the number of faces "*f_count*" which a user wants.
2. Search the Leaf-buffer and the Tree-buffer for the coordinate of vertices associated with *r_level*.
3. Output each triangle consisted of the above coordinate of vertices.

(Pass 1) Calculate the level of detail of *f_count*.

We illustrate a way of calculating *r_level* of the number of faces a user wants in Figure 5. The *r_level* is the degree of the LOD, which is associated with a user-defined number of faces, in an octree. f_count_{max} in Figure 5 is the number of faces constructing the original model.

First, we search *i_level i* which is the maximum number of faces within f_count_i, *i_level i+1* which is the minimal number of faces over f_count_{i+1}. It is easy to set up the number of faces of the simplified model by counting these levels respectively in advance. Next, we calculate *r_level* corresponding to the *f_count*. In general, a model consists of nearly

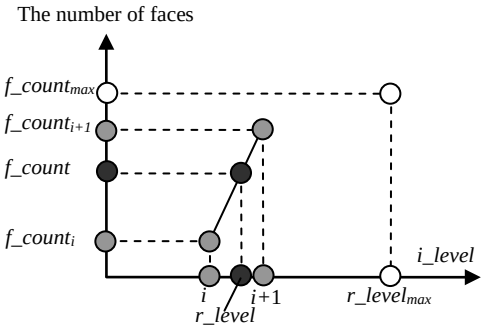


Figure 5. A computation of *r_level*.

uniform faces in 3D-CG. Therefore, we can think that the change in the number of faces is nearly constant, each time a vertex is deleted. We suggest the following equation for calculating the *r_level*.

$$r\_level = i + \frac{f\_count - f\_count_i}{f\_count_{i+1} - f\_count_i} \quad (3)$$

We can generate the simplified models consisting of faces whose number is nearly equal to the user-defined number by using the *r_level*.

(Pass 2) Select coordinate of vertices.

Each coordinate of vertices is selected by *r_level* on a vertex shader of a GPU. We illustrate the selecting way with an example in Figure 6.

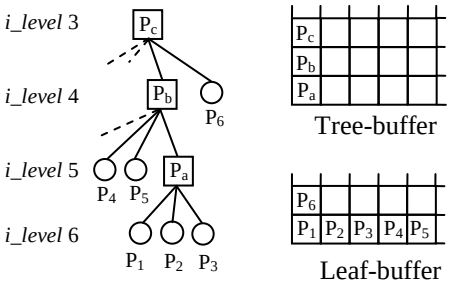


Figure 6. An example of a texture in an octree.

We describe how to calculate a texture coordinate of a parent node associated with a leaf node on the Tree-buffer, when we assume *r_level* is equal to "4.4".

First, an arbitrary leaf node is selected. In this case, we assume that P_1 is selected. When the pixel of P_1 in the Leaf-Buffer is referred, it stored a pointer whose value refers directly to the parent node P_a according to Table 1. We can calculate the texture coordinate of P_a in the Tree-buffer by the pointer.

Table 1. Table of referenced nodes in each leaf node.

Leaf node	$level_m$
P_1	P_a (5.633)
P_2	
P_3	
P_4	P_b (4.382)
P_5	
P_6	P_c (3.457)

Next, we calculate a node associated with *r_level* by nodes whose parent node is located on the same axis of *v*. As described above, all parent nodes which are able to be traced from arbitrary internal nodes are arranged a column of the Tree-buffer texture. Therefore, we calculate the location storing the target node with difference *d* between *r_level* and $level_l$.

$$d = level_l - r\_level \quad (4)$$

where, $level_l$ is level of leaf node.

However, the target node can not always be calculated by equation (4), because both *r_level* and level which internal nodes have are real numbers. Therefore, we consider the following 3 cases according to *d*. These are (Case 1) $d \le 0$, (Case 2) $0 < d \le 1.0$, (Case 3) $d > 1.0$. In (Case 1), the three world coordinates of an arbitrary vertex in the

leaf node are selected, because the level of the leaf node is too small for the *r_level*. In (Case 2), if a level of the parent node for the leaf node is larger than the *r_level*, the three world coordinates of an arbitrary vertex of the leaf node are selected; otherwise, the three world coordinates of an arbitrary vertex of the parent node are selected. In (Case 3), when each level of 2 internal nodes closest to the *r_level* is compared with the *r_level*, the three world coordinates of an arbitrary vertex of the internal node whose level is larger than the *r_level* are selected. In Figure 4, $level_l$ is 6, *r_level* is 4.4. *d* is equal to 1.6 by equation (4). As a result, this is selected as Case 3. The candidate internal nodes are P_a and P_b. The selected node is P_a because *r_level* (4.4) is greater than the level of P_b (4.382).

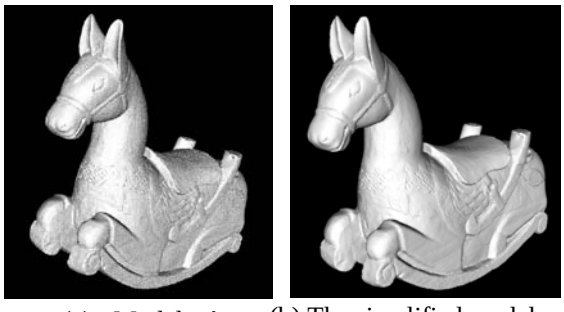
As described above, the access number of times to texture is 3 tops. It is quicker to search in the octree.

(Pass 3) Generate triangles for meshes.

The above process selects each coordinate of vertices for triangles. This pass determines if each triangle is generated in geometry shader. The geometry shader calculates normal vectors of triangles. If the size of the normal vector is not equal to 0, triangle is rendered; otherwise, triangle is not rendered.

3. Experiment and Results

To evaluate the performance of our algorithm, we made 3 experiments. The first experiment is associated with processing time, the second is associated with controlling the number of faces. Figure 7 shows 2 original models and the simplified models whose number of faces are one tenth of original models, respectively. We use "*s_level*" instead of the number of faces, where *s_level* is degree of LOD. The range of *s_level* a user defines is from 1 to 1,000. That is, when *s_level* is 1,000, the number of faces is that constructing the original model. When *s_level* is 1, the simplified model consists of one thousandth the number of faces for the original model. All simplifications were performed on a PC with a Core2 Duo CPU (2.66GHz), 2GB of RAM and an NVIDIA GeForce 8800GTX (768MB) GPU, collected on Windows XP Professional.



(a) Model A (2,208,936) (b) The simplified model of A (226,866)

(c) Model B (3,745,150) (d) The simplified model of B (377,278)

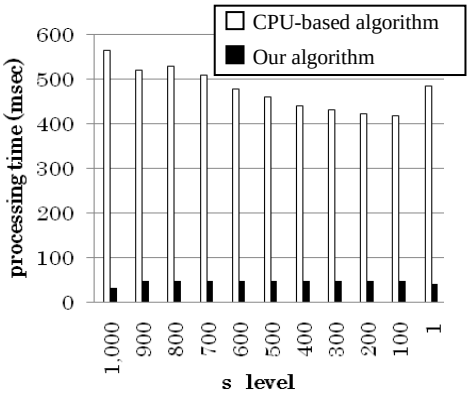
Figure 7. Experimental models and Simplified results.

3.1 Experiment associated with processing time.

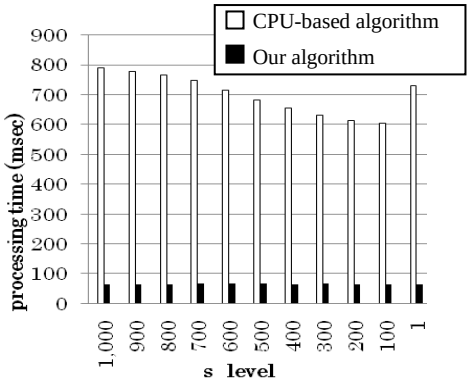
We compare our algorithm with a CPU-based algorithm. The CPU-based algorithm is [Kanaya and Kobori 2002]. Figure 8 shows the comparison of 2 algorithms. The horizontal axis shows the *s_level* and the vertical axis shows processing time. Our algorithm is up to 17 times quicker than the CPU-based algorithm, as shown in Figure 8. Additionally, it is found that the processing time is almost constant at every *s_level*.

3.2 Experiment associated with controlling the number of faces

We verify that when a user defines an *s_level*, the simplified models consist of the number of faces he/she expected. The range of *s_level* a user defines



(a) Processing Time of Model A



(b) Processing Time of Model B

Figure 8. Comparisons of our algorithm and CPU-based algorithm.

is from 1 to 1,000. That is, when *s_level* is 1,000, the number of faces is that constructing the original model. When *s_level* is 1, the simplified model consists of one thousandth the number of faces for the original model. Figure 9 shows a result of model A simplified by using our algorithm. The horizontal axis shows the *s_level* and the vertical axis shows the number of faces.

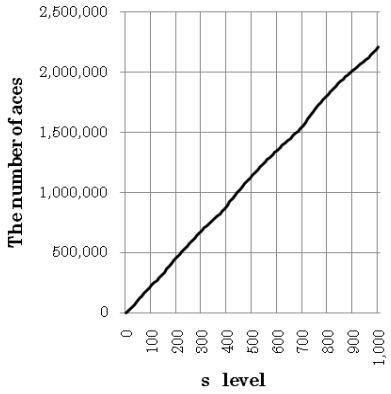


Figure 9. Relationship between *s_level* and the number of faces.

A relationship between *s_level* and the number of faces is almost linearized according to Figure 9. However, the difference between the numbers of faces we have expected and actual results were accurate within 4.7%, because we calculated *r_level* of *s_level* by linear approximation. However, it is easy to get the number of faces a user wants by using our method.

4. Conclusions

We have presented a method for model simplification on the GPU programmable shader and demonstrated how triangle decimation becomes practical for real-time use. We have applied a vertex-clustering algorithm to the GPU and we have presented how to store the clustering information.

We introduced "*r_level*" that is a real number, for the purpose of generating the simplified models which consisted of the number of faces a user expected. We introduced 2 buffers "Tree-buffer" and "Leaf-buffer" for faster access and efficient storage. Additionally, the experiment results showed efficacy.

We have not compared our algorithm with the algorithm based GPU [DeCoro and Tatarchuk 2007]. We would like to compare our algorithm with the algorithm in the future work.

5. REFERENCES

[DeCoro and Tatarchuk 2007] DeCoro, D. and Tatarchuk, N., Real-time Mesh Simplification Using the GPU. Symposium on Interactive 3D Graphics (I3D) 2007, pp. 6, April 2007.

[Garland and Shaffer 2002] Garland, M., and Shaffer, E., A multiphase approach to efficient surface simplification. In VIS '02: Proceedings of the conference on Visualization '02, IEEE Computer Society, Washington, DC, USA, 117–124, 2002.

[Garland 1999] Garland, M., Quadric-based polygonal surface simplification. PhD thesis, Carnegie Mellon University. Chair-Paul Heckbert, 1999.

[Kanaya and Kobori 2002] Kanaya, T., and Kobori, K., A Method of Model Simplification Using Spatial Partitioning. In the journal of the Institute of Image Information and Television Engineers 56(4), 636-642, 2002.

[Lindstrom 2000] Lindstrom, P., Out-of-core simplification of large polygonal models. In SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 259–262, 2000.

[Low and Tan 1997] Low, K.-L., and Tan, T.-S., Model simplification using vertex-clustering. In SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics, ACM Press, New York, NY, USA, 75–82, 1997.

[Luebke and Erikson 1997] Luebke, D., and Erikson, C., View-dependent simplification of arbitrary polygonal environments. In SIGGRAPH ' 97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques, ACM PRESS/Addison-Wesley Publishing Co., New York, NY, USA, 199–208, 1997.

[Rossignac and Borrel 1993] Rossignac, J., and Borrel, P., Multi-resolution 3d approximations for rendering complex scenes. In Geometric Modeling in Computer Graphics, Springer-Verlag, New York, NY, USA, 455–465, 1993.

[Shaffer and Garland 2001] Shaffer, E., and Garland, M., Efficient adaptive simplification of massive meshes. In VIS '01: Proceedings of the conference on Visualization '01, IEEE Computer Society, Washington, DC, USA, 127–134, 2001.

Speeding up probabilistic inference of camera orientation by function approximation and grid masking

Nicolau Leal Werneck
Universidade de São Paulo
Av. Prof. Luciano Gualberto tv. 3, 158
05508-900 São Paulo, SP, Brazil
nwerneck@usp.br

Anna Helena Reali Costa
Universidade de São Paulo
Av. Prof. Luciano Gualberto tv. 3, 158
05508-900 São Paulo, SP, Brazil
anna.reali@poli.usp.br

ABSTRACT

This article presents modifications to an existing technique for camera orientation estimation intending to make it faster for use in real time applications and also for analysis of large image sets. The technique is based on likelihood maximization of a probability function that has the image gradient as the observed data and the camera orientation as parameter values. The camera orientation is inferred from the vanishing points of the image, and the directions of the edges in the environment are assumed to be in three mutually orthogonal directions. The first proposed modification is to substitute the expression that is calculated at each pixel by a computationally lighter approximation. The second proposal is to take in consideration only a few of the pixel lines and columns of the image during the calculations, performing a grid windowing of the image. This article presents the derivation and reinterpretation of the likelihood function approximation and also a performance evaluation.

Keywords

Vanishing point, grid masking, camera orientation, camera localization, Bayesian inference, ML estimation.

1. INTRODUCTION

Camera localization is the Computer Vision problem of inferring the position and orientation of a camera in an environment from one or more pictures captured by it. Camera localization problems are defined by their different restrictions, specially the available data and what parameters are to be estimated. As usual in Computer Vision, it is an ill-posed problem of parameter estimation, and solutions are often based on procedures such as non-linear regression [SW89] and robust estimation [CKY09, HZ03]. One specific case of the localization problem is to estimate just the camera orientation from a single image under the restriction known as "Manhattan World", or also "LEGO Land", that the edges in the environment are in the directions of the coordinate axes. This article presents modifications to existing techniques [CY03, DIM02, SD04, DEE08] that solve this problem using the Likelihood Maximization principle, with a probabilistic observation model where the observed data is the image gradient, and the parameters to be estimated define the camera orientation in the world reference frame. Two modifications are proposed: the substitution of the expression calculated at each pixel by a simpler one, and the use of a grid mask to select pixels. The alternative expression caused great speed gains (60 fold in one test) while exhibiting good convergence. The subsampling technique also caused a 10 fold speed increase with just a 10% reduction of convergence probability in another experiment.

The proposed simplified expression can be seen as the result of a windowing operation by a mask that is calculated from the image gradient norm using a sigmoid function. While the original expression is strictly probabilistic, the proposal is similar to techniques such as Fuzzy Logic and Neural Networks.

In the remainder of this section the problem is further described and previous techniques are briefly reviewed. In Section 2 the existing techniques on which this proposal is based are better explained, and so is the developed technique. This section also brings results of experiments conducted with a database of images with solved orientation parameters to evaluate the proposal. Section 3 brings a few conclusions.

1.1 Problem geometry

The aim of the proposed technique is to obtain an estimate of the spatial orientation of a camera from a single image captured by it. The camera follows the simple *pinhole model* [TV98, chap. 2][HZ03, chap.

6]. In this model there is a camera reference frame whose origin is the focal point of the camera, and the image plane is located at $z = f$, where z is the direction that points outwards from the camera and into the scene. Image coordinates can be converted to this camera reference frame by a linear transformation involving the pixel size and the location of the image origin. The color of pixels are determined by the color of objects that are projected onto the image plane according to the classic perspective transformation [TV98]. The environment is assumed to be composed of rectangular parallelepipeds of faces with different colors and with edges aligned to the coordinate axes of the world reference frame.

The projections of the edges of these objects create edges on the captured images. These image edges typically produce gradient vectors with high magnitude that point to the direction orthogonal to the edge. Image gradients are approximately calculated by linear filters such as the one used in the well known Sobel detector [TV98, chap. 4]. In the present research the Scharr filter was employed [WS02]. The perspective projection causes the well known effect of producing *vanishing points* in the image. Lines that point to the same direction in the environment create either parallel lines in the image, or lines that converge to a vanishing point. The spatial orientation of the camera determines the position of the vanishing points, and the orientation can be therefore estimated from the directions of the image edges [HZ03, sec. 8.6]. This principle is the basis of many different techniques to estimate camera orientation.

1.2 Existing techniques

Many of the existing techniques for vanishing point or camera orientation estimation are either based on the Hough Transform [Shu99, CJRZ10] or on robust estimators [Tar09, Fö10] for matching edges extracted from the image and perform the desired estimation. Extracting edges and defining parameter space accumulators can be a nuisance for some applications, and this is one of the main reasons to look for alternative techniques. It is usually hard to extract edges with good precision, and also to match the edges that refer to the same direction. The technique presented in this article is part of a family of techniques that avoid these problems by using probabilistic models to infer camera orientation directly from pixel values, exploiting the vanishing point restriction.

The probabilistic model is used to perform *maximum likelihood* (ML) estimation to determine the camera orientation $\vec{\Psi}$ from a given input image. The differences between these similar techniques lie in the expression used for the calculation of the image likelihood given $\vec{\Psi}$ and the pixel values, more specifically the image gradient, and in the optimization procedure employed to find the optimal $\vec{\Psi}^*$ that maximizes the likelihood expression.

The directions of the environment edges must be known in order to infer camera orientation from vanishing points. In the present research the orientations are assumed to be in the directions of the coordinate axes, so the edge directions in camera coordinates are easily calculated from the rotation matrix that gives the camera orientation in relation to the world reference frame. Other than camera orientation, most of these techniques can be modified for other tasks such as discovering vanishing points in unknown directions and also estimating intrinsic camera parameters such as the focal distance f.

2. METHODOLOGY

This section brings more detailed explanations about how the existing and the proposed techniques work. They are all procedures that create an estimate of a camera orientation $\vec{\Psi}$ from a given input image. This orientation is a rotation matrix in three dimensions, and as such can be parameterized in different ways. The most popular alternatives are Euler angles, the Rodrigues formula, and quaternions, which are used in this work. But this representation is not relevant for the following subsections, where the reader can just assume $\vec{\Psi}$ is given as a 3D rotation matrix.

The next subsection describes the original probabilistic technique for estimating camera orientation from image gradient [CY03] and some modifications. The following subsection brings the new proposals. These techniques are all based on the Maximum Likelihood principle. They are more specifically *maximum a posteriori* (MAP) estimators, that can be seen as regularized ML estimators. These methods need a function called observation model, which is a conditional *probability density function* (PDF) of observing a measured data set given certain condition parameters. This function is used as a likelihood function, where the observed data is taken from the image gradient, and the conditional parameters are the camera orientation (related to the vanishing points locations), image coordinates of each pixel, and a pixel class that will be explained below. Once the expression is defined and the data collected, an optimization technique is used to find the parameters that maximize this MAP estimator. The $\vec{\Psi}^*$ found by this optimization is the desired camera orientation estimate.

2.1 Original observation model

In the first observation model proposed related to our technique [CY03] the likelihood of the whole image is factored as the product of the likelihoods of the

gradients $\vec{E}_{\vec{u}}$ at each pixel $\vec{u}$. These individual PDF are also further factored as products of the likelihoods of the gradient norms $E_{\vec{u}}$ and edge angles $\phi_{\vec{u}}$ yielding

$$P(\vec{E}_{\vec{u}}|m_{\vec{u}}, \vec{\Psi}, \vec{u}) = P(E_{\vec{u}}|m_{\vec{u}})P(\phi_{\vec{u}}|m_{\vec{u}}, \vec{\Psi}, \vec{u}). \quad (1)$$

The edge angle $\phi_{\vec{u}}$ is orthogonal to the gradient direction $\angle\vec{E}_{\vec{u}}$. The formula has two important characteristics. The first is that the PDF of the gradient norm $E_{\vec{u}}$ depends only on the pixel class $m_{\vec{u}}$. This class can be one of five possibilities: class 1 means the pixel is not an edge, classes 2–4 are edges on each of the three coordinate axes of the world reference frame and class 5 is a non-aligned edge.

The second characteristic is that the PDF of $\phi_{\vec{u}}$ also depends on $m_{\vec{u}}$, but also on the camera orientation $\vec{\Psi}$ and the coordinates of pixel $\vec{u}$. When $m_{\vec{u}} = 1$ or 5 we assume all gradient directions are equally probable, so $P(\phi_{\vec{u}}|m_{\vec{u}}, \vec{\Psi}, \vec{u})$ becomes a uniform distribution in these cases. For $m_{\vec{u}} = 2$, 3 or 4 we calculate the probability of the measured direction. For that we first use $\vec{\Psi}$ to calculate $\vec{r}^m = (r_x^m, r_y^m, r_z^m)$, a vector in the direction of the edges of the class $m_{\vec{u}}$ in the camera reference frame. The location where a line extended from $\vec{r}^m$ crosses the image plane is the vanishing point. The vector can also be parallel to the plane, in which case there is no actual vanishing point but it is still possible to calculate the directions of the edges. The direction on each pixel is:

$$\vec{\theta}_{\vec{u}}^m = \left(\frac{r_x^m}{r_z^m}f + u_x, \frac{r_y^m}{r_z^m}f + u_y\right). \quad (2)$$

One way to calculate $P(\phi_{\vec{u}}|m_{\vec{u}}, \vec{\Psi}, \vec{u})$ used in previous techniques is to determine the vanishing point direction angle $\angle\vec{\theta}_{\vec{u}}^m$, then subtract it from the edge direction $\phi_{\vec{u}}$, and this difference is then used as parameter to the PDF of the observation error in the measured edge directions. This PDF has been assumed in previous works to be uniform [CY03], triangular [DIM02], Gaussian [SD04] and a Generalized Laplace distribution [DEE08].

Two different PDF are used to implement $P(E_{\vec{u}}|m_{\vec{u}})$. For $m_{\vec{u}} = 1$, $P_{off}(E_{\vec{u}})$ is used, and $P_{on}(E_{\vec{u}})$ is used otherwise. Different assumptions have been made about these functions too. Both measured values [CY03, DIM02] and Gaussian models [SD04] have already been used.

As previously mentioned, the likelihood of the complete image is a product of terms given by Equation 1. This product can be used to define a ML estimator, but what is usually done is to improve it by using the information of *a priori* probabilities of $P(m_{\vec{u}})$, to define a MAP estimator. The logarithm of the resulting expression is also taken to replace the product by a summation, what does not change the location of the maximal points. Considering all this, and using M^k for $P(m_{\vec{u}} = k)$, Φ^k for $P(\phi_{\vec{u}}|m_{\vec{u}} = k, \vec{\Psi}, \vec{u})$ we arrive at the expression:

$$L\left(\vec{\Psi}\right) = \sum_{\vec{u}} \log\Big(P_{off}(E_{\vec{u}})\Phi^1 M^1 + P_{on}(E_{\vec{u}})\Phi^5 M^5 + P_{on}(E_{\vec{u}})\textstyle\sum_{k=2}^{4}\Phi^k M^k\Big) \quad (3)$$

The camera orientation estimate is therefore the rotation $\vec{\Psi}^*$ that maximizes the function L. In the original proposal the summation is performed over all the image pixels [CY03], but just like with the PDF definitions, other researchers have proposed different ways to select subsets of the image pixels over which the summation should be performed, hoping to make the calculation faster and also smooth the estimator function. One proposal is to divide the image in square tiles, and sample a single pixel randomly from each one, a different pixel at each calculation [DIM02]. Another possibility is to select only a few of the pixels with the largest values of $E_{\vec{u}}$ [SD04]. The probabilistic modeling of the pixel being or not on an edge can be even dropped and substituted by the use of an edge-finding algorithm [DEE08]. In this case the argument of the log in Equation 3 becomes simply $\sum_{k=2}^{5}\Phi^k M^k$, but this technique depends on an initial edge extraction, that did not exist in the original proposal.

Other aspects where the techniques differ is the application of the Expectation-Maximization algorithm, where values for M^k are also iteratively estimated [SD04, DEE08], and the optimization algorithms used. Alternatives range from coarse-to-fine search at regularly sampled points [CY03], stochastic importance sampling [DIM02], and continuous non-linear optimization methods [SW89] such as Levenberg-Marquardt [SD04] and BFGS [DEE08].

2.2 Function approximation

This subsection describes the first major modification investigated in this research, which is substituting the original arithmetical expression for the likelihood function by a computationally simpler approximation. The following subsection covers the use of a grid mask to select the pixels to be considered in the calculations.

Tests performed with an implementation of the original likelihood expression (Equation 3) revealed that much of the computation time was spent on functions to compute the logarithm and the arc-tangent used to calculate $\angle\vec{\theta}_{\vec{u}}^m$. A removal from the program of the procedure calls related to these operations, while keeping all the rest of the

calculations, resulted in an approximately ten fold speed gain, showing experimentally that avoiding these operations can be a good strategy to reduce the calculation time. Arc-tangent was the most costly operation of the three, considering both the time of a single calculation and the number of calls at each calculation iteration in the summation loop.

The modifications begin by replacing the logarithm with the first-order approximation

$$\log(b+a) \approx \frac{a}{b} + \log(b), \tag{4}$$

where a represents the terms that depend on $\vec{\Psi}$, and b the terms that remain constant during the optimization. The $\log(b)$ term can therefore be ignored as it does not influence the solution, and the resulting approximation becomes

$$\sum_{\vec{u}} W'(E_{\vec{u}}) \sum_{k=2}^{4} \Phi^k \frac{M^k}{\Phi^1}, \tag{5}$$

where the *mask generating function*

$$W'(E_{\vec{u}}) = \left(\frac{P_{off}(E_{\vec{u}})}{P_{on}(E_{\vec{u}})} M^1 + M^5 \right)^{-1}. \tag{6}$$

The function W' produces, at least with the appropriate parameters, a sigmoid curve, similar to the logistic or to the hyperbolic tangent functions. The second approximation used was to replace this function by W, the logistic function applied to $E_{\vec{u}}$ translated by p_1 and scaled by p_2

$$W(E_{\vec{u}}) = \left(1 + e^{-p_2(E_{\vec{u}} - p_1)} \right)^{-1}. \tag{7}$$

Replacing W' for W at Equation 5 and ignoring the constant M^k/Φ^1, that only scales the function, finally produces the proposed estimator:

$$\tilde{L}\left(\vec{\Psi}\right) = \sum_{\vec{u}} W(E_{\vec{u}}) \sum_{k=2}^{4} \Phi^k, \tag{8}$$

Figure 1 displays, at the top, the probability models of the gradient magnitudes with measured values, provided by the authors of [CY03], and also the Gaussian models from [SD04] (mean 8.28 and standard deviation 6.21 for P_{on}, and respectively 1.13 and 0.77 for P_{off}). On the bottom of the figure, the continuous and dashed curves are W' obtained from the two PDF models mentioned, and the red dotted curves are W with two different sets of parameters ($p_1 = 10$ $p_2 = 0.4$ and $p_1 = 3.1$ $p_2 = 3.0$).

Figure 2 shows an image from the YorkUrbanDB image database [DEE08]. This image set has 102 indoor and outdoor images of man-made environments, and the orientation of each image was

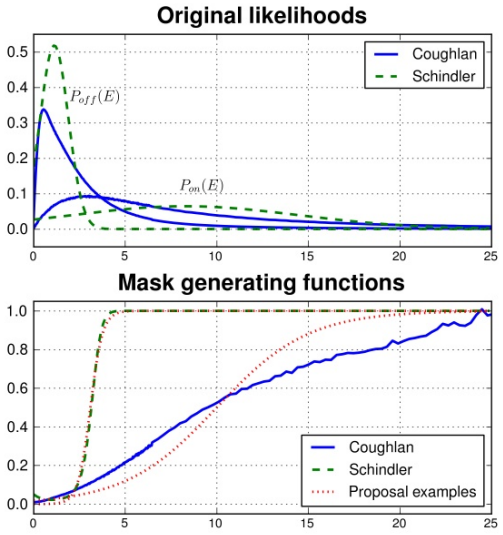


Figure 1: Original gradient magnitude likelihood functions, resulting mask generating functions and examples of the proposed function.

obtained from edges and with a manual labeling process. Intrinsic parameters of the camera are also provided, enabling interested researchers to test their techniques and compare to others. Possible radial distortions of the images were not taken in account in this work, but the projection center coordinates and focal distance that are provided were used.

The leftmost graphic of the figure displays the input image. The next one displays the values of W calculated over each pixel, with white representing the zero level, ($p_1 = 20$ and $p_2 = 0.2$ were used). The two graphics to the right display the horizontal and vertical components of the normalized direction vector. The red color denotes negative values, but even in a monochromatic mode it is possible to see how edges in the direction of the derivative vanish on each graphic. The edge mask obtained with W has been applied to these gradient images, clearing out the noise that would be otherwise noticeable in the large white areas of these images.

In the program created to implement this expression the edge mask is calculated and stored in memory before the optimization procedure starts, so only memory accesses are needed to obtain the values during the calculations. Something similar can be done with other techniques, because $P(E_{\vec{u}}|m)$ does not depend on Ψ, only Φ^m does.

The last modification done to the likelihood expression was to substitute the calculations of arc-tangents by dot products. Instead of calculating the angles of the gradient and vanishing point directions, these vectors are simply normalized and multiplied by each other. Because the gradient is orthogonal to the edge direction, this multiplication

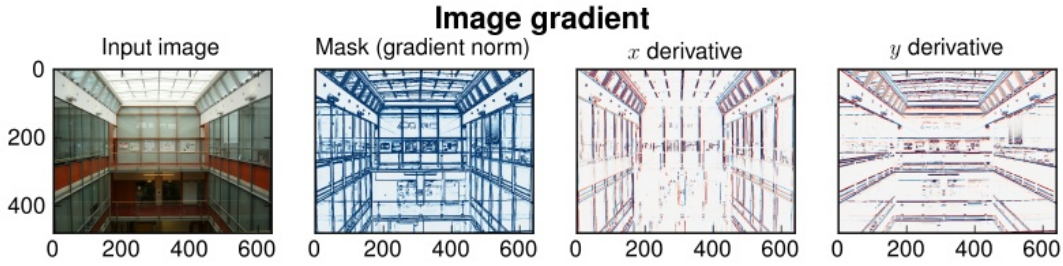


Figure 2: Gradient of an YorkUrbanDB image. The second image is the edge mask calculated from the gradient vectors absolute values. The two rightmost graphics are the masked gradient x and y components.

yields $\gamma^m = \sin(\phi_{\vec{u}} - \angle\vec{\theta}^m_{\vec{u}})$. This is a good approximation of the identity function for small values, so the product result can be directly used in the angle error PDF. The function used was therefore:

$$\Phi^m = \begin{cases} \frac{2}{p_3}\left(1 - \frac{|\gamma^m|}{p_3}\right) & \text{if } |\gamma^m| < p_3 \\ 0 & \text{if } |\gamma^m| \geq p_3 \end{cases}, \quad (9)$$

where we note that the $2/p_3$ multiplication can be dismissed without affecting the optimization results.

The normalization of $\vec{E}_{\vec{u}}$ and $\angle\vec{\theta}^m_{\vec{u}}$ can be performed quickly using a special `rsqrt` instruction available in many modern processors that calculates an approximation of the reciprocal of the square root of numbers. This instruction was used in the implementation tested, and so were SIMD (*single instruction, multiple data*) instructions that allow calculations to be performed simultaneously both for the three vanishing points, and also for the three image channels when possible. The three image channels were independently considered in the calculations, with just the pixel coordinates and $\vec{\theta}_{\vec{u}}$ in common. The final likelihood value is therefore the summation of likelihoods for each channel.

The program was implemented using `Cython` [Sel09], with a few routines implemented in C in order to make use of the special processor instructions mentioned. Another implementation was made based on [CY03], using arc-tangent and logarithm calls inside the loop, but with some similarities to the implementation of the proposal, such as using SIMD instructions for some operations, and caching constant values.

Tests were performed with the YorkUrbanDB images at different values of Ψ to measure the speed of the proposed function relative to this implementation of the original. Speed gains from 50 up to 64 times were found in one computer (`c1.xlarge` instance from Amazon Web Services [Ser]), where the mean time to calculate the likelihood of one image using the classic function was 1.10 ± 0.06s versus 18.9 ± 2.4ms for the proposed algorithm. Although these numbers naturally varied according to the processor employed, accelerations of more than 10 times were often detected in other tests.

The positive impact of these function modifications on the calculation speed is not surprising. But the impact of these modifications on the performance of the optimization procedure must be now studied to validate the proposed technique. This analysis will be presented in Subsection 2.4. But it should be noted that this proposed modification did not intend to numerically approximate the original likelihood function values. The original function serves more as a theoretical foundation, and the modifications do not seek to approximate it exactly, but only retain characteristics such as the positions of the extremal points and gradient directions.

When the logarithm of the likelihood is used instead of the original function in an optimization, the produced function does not approximate the original numerically, but is still useful for the optimization. So the performance of such modifications should not be measured by looking at approximations errors, but at the optimization results instead. In the same way, because the modifications proposed here include dropping some constant terms, the resulting function cannot be compared to the original function, so no error analysis was performed, only performance analysis of the optimization procedure. Despite of that, the modifications are in fact initially based on first-order approximations of the original function, justifying the use of the term *approximation*, even though the final proposed function does not approximate the original one numerically.

The proposed function also differs from the original in that the parameters of the mask generating function are only indirectly related to the gradient norm probabilities. While it is possible to fit the parameters to a mask function taken from histograms, it is better to look for parameters that maximize the performance of the final optimization procedure. The sensitivity of the performance to these parameters, and also to the gradient norm probabilities is a topic that the proposed modifications bring up, but was not studied here.

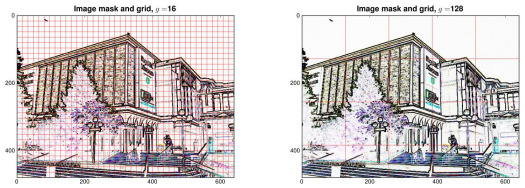


Figure 3: Two example grids with spacing equal to 16 and 128 over the norm of the gradient of an example image.

2.3 Grid mask

A sampling strategy was introduced to reduce the number of pixels used in the calculations, and also try to make the estimator smoother. The strategy is to select only a few of the pixel lines and columns of the image. These were selected at regular intervals, with the spacing controlled by the parameter g. The result is the same of applying a mask to the image with the shape of a square grid, with a continuous line or column at every g pixels, starting from the image origin. When $g = 1$ all pixels are used. When $g = 2$, $3/4$ of the pixels are used, and $7/16$ when $g = 4$.

The grid masking is proposed here as a theoretically more suited way to subsample images when edges are the target features. Other subsampling techniques simply regard edges as a kind of pixel, and not as a geometric entity without area. Some authors quote statistics such as "10% of the image pixels are edges", but such statements miss some important points about image edges. The number of pixels of an image increases with the square of resolution, but the number of pixels that lie over and edge should increase linearly. New edges may be introduced as resolution increases, affecting positively this proportion of edge pixels to image size, but the relative number of pixels of an existing edge still decreases linearly with resolution, and subsampling strategies should take this effect in consideration.

As image resolution increases the number of edge pixels found over a grid line or column should remain constant, while non-edge pixels increase linearly with resolution — assuming that no new edges are introduced, and that edge directions are not exactly aligned to the grid. One interesting characteristic of grid masking is that if the edges have a minimum length, the grid spacing can be made small enough as to guarantee that a minimum number of points over any edge in the images is sampled. Grid masking also avoids sampling groups of neighboring pixels, what is generally thought to be good because pixels are assumed to be independent in the probabilistic model. Figure 3 shows the grid lines and columns overlaid to the gradient of the three channels of an image from the YorkUrbanDB database. The top graphic has the grid spacing parameter $g = 16$, and the lower $g = 128$.

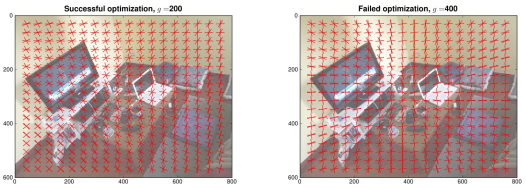


Figure 4: Successful and failed optimizations.

2.4 Optimization

With the likelihood function and sub-sampling technique defined, an optimization technique can now be used to produce orientation estimates from input images. The algorithm used was the modified Powell's method from `SciPy` [JOP$^+$]. Figure 4 displays a successful optimization, obtained with $g = 200$, and a failed one with $g = 400$. This is an 800x600 pixels image captured with a consumer digital camera. Line segments in the directions of the three vanishing points obtained form the solution were plotted in regularly spaced points over the images, and it is possible to see how the edges are aligned to the objects in the environment. In the failed optimization the solution found was not much far from the initial condition, which was no rotation.

The parameterization used for the rotations was quaternions. The vector $\vec{\Psi}$ has three dimensions, and are the three quaternion parameters that are directly related to the direction of the rotation axis. The fourth parameter, related to the rotation angle, is calculated as $\sqrt{1 - ||\vec{\Psi}||^2}$. If $||\vec{\Psi}|| > 1$ no quaternion can be directly produced. In this case the quaternion is obtained from $-\vec{\Psi}/||\vec{\Psi}||$. It should be noted that no symmetries were taken in account in this parameterization, so multiple $\vec{\Psi}$ values are equally acceptable solutions, and can be obtained from each other by 90 degrees rotations around the axes. The problem of associating the axes properly, when possible, was not considered in this research.

As in previous works, optimization is initiated from different starting points [DEE08], although only two were used in the present experiments. One point considers no rotation, and the other a 45 degrees rotation around the vertical axis. This explores the tendency of the camera to be upright, and the ambiguity resulting from 90 degree turns. After the two optimizations are performed, the solution with the highest likelihood is picked as the best estimate.

Figure 5 shows an evaluation of this optimization for different grid spacings g. The parameters used for $\tilde{L}$ in this experiment were $p_1 = 20$, $p_2 = 0.2$ and $p_3 = 0.1$. There is a compromise between calculation speed and the quality of the solutions obtained. The decreasing green curves show the probability p, estimated from the N=102 images, of the obtained

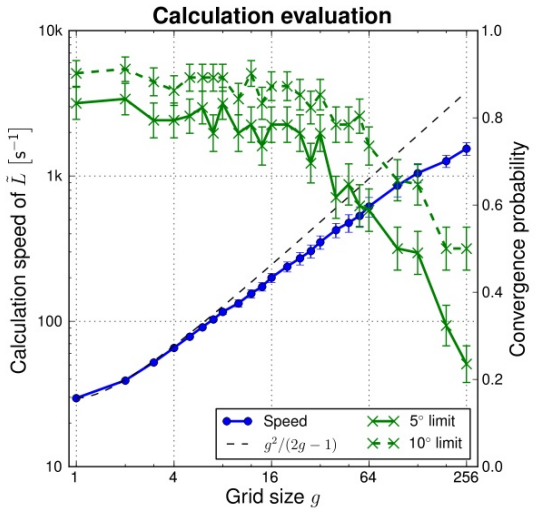


Figure 5: Speed of the calculations as a function of the grid spacing, and two quality evaluations.

solution lying at 10° or 5° of the orientation provided in the database [DEE08]. The 10° curve is naturally above the 5° one. The uncertainty interval plotted is a single standard deviation above and below the points, calculated as $\sqrt{p(1-p)/N}$.

The speed estimate is the number of times $\tilde{L}$ can be calculated per second as a function of g. To calculate this, the elapsed time and number of function calculations performed were first stored for both optimization runs of all test images. The speed was then estimated for each optimization run by dividing the number of calculations by the elapsed time. The mean number of iterations was 276 for all optimization trials with all g values, with a 89.3 standard deviation. The increasing blue continuous line in the graphic is the mean and 6σ interval of this speed for all optimizations performed for each g. It should be noted this experiment was performed in a slower machine compared to the one used for the measurement reported in Subsection 2.2.

The increasing black dashed line in Figure 5 shows how speed should increase if it depended only on the reduction of the number of pixels, where speed gain should be $g^2/(2g-1)$. The smaller speed values that were measured are coherent with the addition of a constant time to the calculation time, the reciprocal of the speed value.

This analysis only considers the individual performance of the proposed target function and the effects of the grid sampling. Another test was performed in order to validate the proposed function. The objective was to find out if the modifications were causing the extremal points to be located in positions further from the true solutions than the points produced by the original function.

To perform this test the solution found with the proposed method was used as initial estimate for a second optimization on the original function. The error of the first and the second optimizations compared to the estimate in the database were then analyzed. The modifications would be considered destructive if the errors in the first optimization were higher than the errors from the second, i.e. the second optimization would "fix" the first one. On the other hand, if the modifications are acceptable the second optimization should not improve the solutions much.

The result was that from the 102 YorkUrbanDB images 53 had their errors reduced after the second optimization. From these, 5 were improvements from more to less than 10° away from the correct solution. On the other hand, from the 49 cases where the second optimization ended with a larger error, there were 6 cases where the initial solution was below 10° but the second was beyond. So there is no indication that using the original expression can be critical to improve the performance obtained with the proposed function, at least with the optimization algorithm that was used and with no subsampling performed.

3. CONCLUSION

This article demonstrated modifications made to existing techniques for camera orientation estimation to attain higher calculation speed. The techniques are based on the optimization of a MAP estimator that has the image gradient values as observed data, and the camera orientation as estimated parameter. It works by finding the orientation that causes the best alignment of the image gradient to the vanishing points created by the directions of the three mutually orthogonal axes of the world reference frame.

The original expression to calculate the likelihood was modified by an approximation that avoids the calculation of arc-tangents by using dot products, and also replaces the logarithm of a summation at the expression for each pixel by a summation where all the terms are strictly dependent on the gradient directions and camera orientation. These pixel summations are weighed in the total image summation by a coefficient calculated by applying a sigmoid function to the gradient norms.

This coefficient takes the role performed originally by the likelihoods P_{on} and P_{off}, and also the *a priori* probabilities M^1 and M^5. The need to measure these parameters is replaced by having to choose just p_1 and p_2. The third parameter p_3 shapes the likelihood of gradient directions. More tests still have to be conducted to determine the best parameters, but the technique seems to be robust to variations on them. Outside of these parameters, the other parameters that

must be set in order to use the technique are the ones related to the optimization.

A grid masking technique was also proposed to select a subset of the image pixels to take in consideration in the calculations. It was inspired in the usual curve tracking technique of searching for edges over spaced lines normal to an initial estimate of the curve location[BI98, chap.5], and also on the Canny edge extractor [TV98]. It subsamples the image in a deterministic and more reliable way, and has been proven effective.

Some planned extensions to this research are to better choose the function parameter values and turn the grid masking into a search of maximal points of the derivative in the direction of the line or column. The gradient calculations can also be restricted to the grid vicinity to speed up calculations. Other subsampling techniques can also be applied together with a grid mask. For example, random sampling could be performed only within the mask pixels, or a random sampling could be performed in the whole image initially, but instead of picking just a single pixel from each trial, picking a whole group of pixels inside a cross or square mask centered at each generated pixel.

This fast orientation estimation algorithm is planned to be used in real time to track the orientation of a camera with a Kalman filter or a similar technique. An attempt will be made to reuse the data remaining from the grid masking to also extract edges. The resulting edge observations will be fed to a monocular simultaneous localization and mapping (SLAM) system [NDL08] that exploits the restrictions on the edge directions.

ACKNOWLEDGEMENTS

The authors thank the support from Capes, CNPq (Proc. N. 475690/2008-7 and N. 305512/2008-0) and FAPESP (Proc. N. 2008/03995-5).

REFERENCES

[BI98] Andrew Blake and Michael Isard. *Active Contours*. Springer, 1998.

[CJRZ10] Xuehui Chen, Ruiqing Jia, Hui Ren, and Yinbin Zhang. A new vanishing point detection algorithm based on hough transform. *Computational Sciences and Optimization, International Joint Conference on*, 2:440–443, 2010. doi:http://doi.ieeecomputersociety.org/10.1109/CSO.2010.163.

[CKY09] Sunglok Choi, Taemin Kim, and Wonpil Yu. Performance evaluation of RANSAC family. In *20th British Machine Vision Conference*, 2009. Available from: http://www.bmva.org/bmvc/2009/Papers/Paper355/Paper355.pdf.

[CY03] James Coughlan and Alan Yuille. Manhattan world: orientation and outlier detection by bayesian inference. *Neural Comput.*, 15(5):1063–1088, 2003. Available from: doi:10.1162/089976603765202668.

[DEE08] Patrick Denis, James H. Elder, and Francisco J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. In David A. Forsyth, Philip H. S. Torr, and Andrew Zisserman, editors, *ECCV (2)*, volume 5303 of *Lecture Notes in Computer Science*, pages 197–210. Springer, 2008.

[DIM02] Jonathan Deutscher, Michael Isard, and John Maccormick. Automatic camera calibration from a single manhattan image. In *Eur. Conf. on Computer Vision (ECCV)*, pages 175–205, 2002.

[Fö10] Wolfgang Förstner. Optimal Vanishing Point Detection and Rotation Estimation of Single Images of a Legolandscene. In *Int. Archives of Photogrammetry and Remote Sensing*, pages 157–162. ISPRS Symposium Comm. III, Paris, 2010.

[HZ03] Richard Hartley and Andrew Zisserman. *Multiple View Geometry*. Cambridge Univ. Press, 2nd edition, 2003.

[JOP+] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. Available from: http://www.scipy.org/.

[NDL08] José Neira, Andrew J. Davison, and John J. Leonard. Guest editorial special issue on visual slam. *Robotics, IEEE Transactions on*, 24(5):929 –931, oct. 2008. doi:10.1109/TRO.2008.2004620.

[SD04] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *CVPR (1)*, pages 203–209, 2004.

[Sel09] Dag Sverre Seljebotn. Fast numerical computations with cython. In Gaël Varoquaux, Stéfan van der Walt, and Jarrod Millman, editors, *Proceedings of the 8th Python in Science Conference*, pages 15–22, Pasadena, CA USA, 2009. Available from: http://conference.scipy.org/proceedings/SciPy2009/paper_2.

[Ser] Amazon Web Services. Available from: http://aws.amazon.com.

[Shu99] Jefferey A. Shufelt. Performance evaluation and analysis of vanishing point detection techniques. *PAMI, IEEE Transactions on*, 21(3):282–288, 1999. doi:10.1109/34.754631.

[SW89] George A. F. Seber and Christopher J. Wild. *Nonlinear Regression*. John Wiley & Sons, Inc., 1989.

[Tar09] Jean-Philippe Tardif. Non-iterative approach for fast and accurate vanishing point detection. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 1250 –1257, sep. 2009.

[TV98] Emanuele Trucco and Alessandro Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

[WS02] Joachim Weickert and Hanno Scharr. A scheme for coherence-enhancing diffusion filtering with optimized rotation invariance. *Journal of Visual Communication and Image Representation*, 13(1-2):103 – 118, 2002. Available from: doi:10.1006/jvci.2001.0495.

Off-line Handwritten Arabic Words Segmentation Based on Structural Features and Connected Components Analysis

Moftah Elzobi, Ayoub Al-Hamadi, Zaher Al Aghbari*
Institute for Electronics, Signal Processing and Communications
Otto-von-Guericke-University Magdeburg, Germany
**Department of Computer Science, University of Sharjah, UAE*
{*Moftah.Elzobi, Ayoub.Al-Hamadi*}*@ovgu.de*

***Abstract*— A precise and efficient segmentation for handwritten Arabic text is a vital prerequisite for the accuracy of the subsequent recognition phase. In this paper, we present a dual-phase segmentation approach. The proposed approach starts first by detecting and resolving sub-words overlapping, then a topological features based segmentation is applied by means of a set of heuristic rules. Because of its crucial importance, the segmentation phase is preceded by a handwritten specific pre-processing phase, that considers issues like word's skew- and slant- correction. The proposed approach has been successfully tested on a database of handwritten Arabic words, that contains more than 3000 words images. The results were very promising and indicating the efficiency of our approach.**

***Keywords*- Arabic Handwriting Segmentation, Handwriting Topological Features, Pattern Recognition.**

I. INTRODUCTION

PEOPLE nowadays expecting that, modern as well as historical human knowledge and cultural resources are digitally available as electronic text, which can be fast, efficiently and easily accessed. Resources that are not converted properly to digital text, e.g., Unicode, ASCII, and etc., soon will become obsolete or even inaccessible for researchers, scholars and general public. This means lose of an important and huge amount of the human cultural memory.

Off-line Optical Character Recognition (OCR) is the technological means used for converting handwritten, typewritten and printed text into a digital text. Latin alphabet based and Chinese characters based scripts have been the subject of extensive research since decades, that lead to significant achievements in the field [1], [2]. Despite the fact of being the world second most used alphabet, reported works that address the off-line OCR issues related to Arabic alphabet based scripts, e.g., Arabic, Persian, Urdu, Ottoman, Kurd, and etc., are relatively less in quantity and quality.

The complexities that hindering fast progress in this research field can be related to the cursive written form of Arabic script and to the high variability of character's forms with regard to their positions (isolated, begin, middle and end) within a word, that makes an efficient segmentation hard and error-prone process. It is also observed that techniques proved successful for Latin and/or Chinese, cannot be directly applied without fundamentals changes [1], [3], [4], [5], [6].

In this paper, we introduce a topological feature based segmentation approach for handwritten Arabic words. In order to compensate limitations of extraction steps such as binarization, our methodology starts by pre-processing steps such as small holes filling and smoothing the out-cropping pixels. Then to reduce the extreme variability of handwritten words, normalization issues such as Slant- and Skew- correction is considered. The segmentation then conducted through a dual-phase procedure. In the first phase, a connected component analysis is performed, in order to resolve sub-words overlapping. Then topological features based segmentation is carried out to segment the word into identifiable units representing their constituent characters.

In literature relatively few works are addressing the problem of Arabic text segmentation. Hereafter we will briefly discuss the most important published related works. In [3] a recognition system for off-line cursive handwriting is presented. The system is segmentation based one, thinned and smoothed images of the strokes are processed and two representations for each stroke are generated. The first representation is a direct straight-line approximation. The other is what they called a reduced graph with loops reduced to vertices, then temporal information of the strokes is extracted by following their straight-line representation from right to left. Finally, cursive stroke representatives are segmented to small parts called tokens that passed to the recognizer.

In an attempt to avoid over-segmentation, [7] propose an analytical segmentation approach, that is trying to extract the whole stroke that represents the character by means of the so called Character Key Feature Set (KF), which is the set of End-Points, Branch-Points, Loop-Points and Dot-Points from the word thinned image. First, the minima and maxima

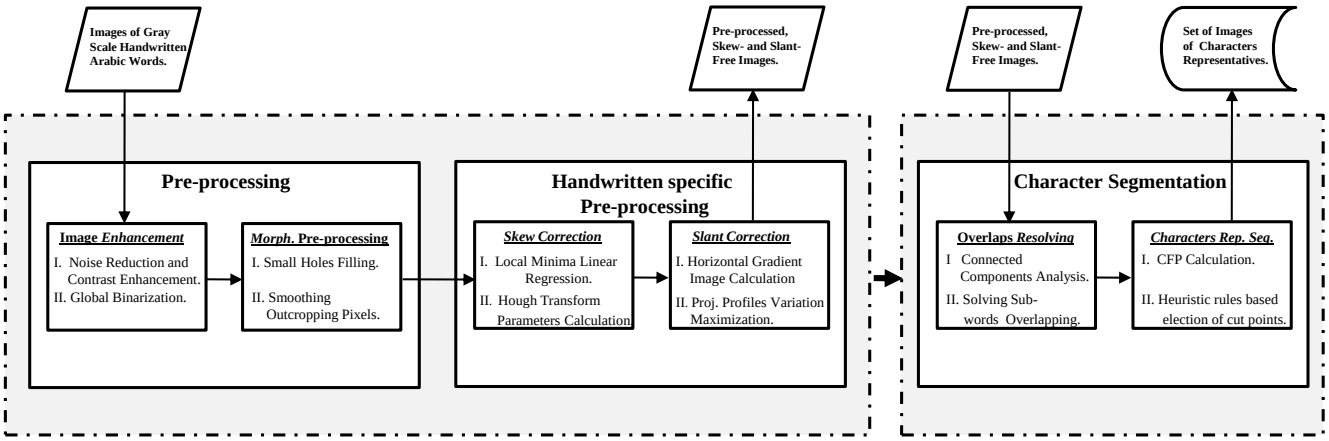


Figure 1. Diagram of the proposed Methodology

of the thinned image are calculated, then the so called Key Features Segments (KFSg) are determined. Secondly, A set of heuristic rules that employ KF set, are applied on the set of the minima in order to elect cut candidates among them.

Ref. [8], proposed an approach, in which a tentative over-segmentation is first performed on the text image, the result is a set of what they called "graphemes", the approach differentiates among three types of graphemes. The segmentation decisions are confirmed upon the recognition results of the merged neighboring graphemes; if recognition failed another merge will be tried until successful recognition.

Ref. [4], presenting an algorithm for printed text segmentation, in which the vertical projection histogram of each line of the source binary image is computed, which then processed to generate a string indicating relative variations in pixels. Finally, a search for patterns in variations is conducted in order to segment the characters' representatives.

II. Methodology

Our methodology consists mainly of two phases, in the first phase the issue of pre-processing is considered. In which traditional pre-processing steps e.g. filtering, binarization, etc., as well as Handwriting specific issues like skew correction and slant correction are conducted. The second phase is the segmentation, which performed in dual-phase procedure.

Given the fact that most Arabic words are consisting of multiple sub-words [9], those are both specially disconnected and vertically overlapping, connected components based analysis and subsequently resolving of sub-words overlapping, are vital for the following character segmentation phase. Ultimately, a topological features based heuristics are applied in order to segment the words into their constituent character representatives. Fig.1 is depicting the proposed methodology.

A. Pre-processing

The words images that we experimented on are gray scale images, taken from an under construction database; conventional flatbed scanner is used to extract the text with 350 dpi resolution. To suppress noisy pixels, whilst preserving edges a median filter is applied on the gray scale images [10]. Then a global threshold (Otsu's method based) is used to produce binary versions. As a consequence of the extraction and binarization processes, issues like smoothing out outcropping pixels and small holes filling should be dealt with. Morphological based operations such as $Close$ and combination of $Open$ and $Reconstruction$ are employed respectively to solve those issues.

To reduce the amount of information to be processed, to the minimum necessary for conducting our segmentation, and also to ease the process of extraction the critical features points, thinning operation is applied on the enhanced binary words' images. The thinning approach that we adapted is based on the Zhang-Suen's thinning algorithm [11]. In the following two subsections, we will discuss and suggests improvements for two off-line handwritten specific pre-processing issues, namely skew- and slant- correction.

1) Skew correction and baseline estimation: Skew correction and baseline discovering is proven to be of critical importance for segmentation of handwritten Arabic text. Various techniques have been reported in literature; each with its pros and cons [12], [13]. Hough transform (HT) is one of such methods, that is relatively insensitive to noise and tolerates gabs within Arabic words [14]. As for the baseline detection HT is insensitive to line direction. Consequently, it performs badly when the longest stroke is not parallel to the word baseline.

Another method is based on the linear regression of local minima of the word image skeleton (LMR) [15]. Benefiting from the fact, that most of local minima (LM) points are usually occurring on, or near of the baseline; the problem of

finding the baseline can be reduced to a linear fitting problem of local minima points. Even though LMR is not as accurate as HT, its main advantage is the relatively insensitivity to strokes' direction that is not parallel to the baseline.

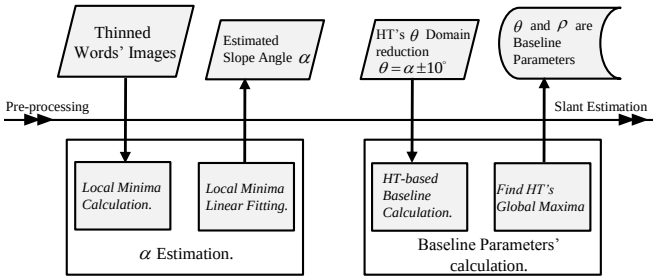


Figure 2. Diagram illustrates the proposed steps for words baseline estimation

Experimentally, we noticed that reducing the Domain of HT's θ parameter according to a priori direction estimation, firstly, increases accuracy, and secondly, reduces the computation power needed. Thus we propose a HT based technique combined with a LMR, for baseline estimation. The LMR is used for a priori estimation of the HT's θ parameter. Fig. 2, shows a diagram depicting the propose approach for baseline estimation.

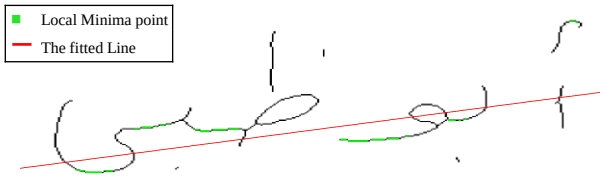


Figure 3. A thinned text image with all possible LM points and their correspondence fitting line.

The first step in the proposed technique starts by calculating the fitting line of LM points according to Eq.1,2, and 3.

$$y = a + bx \tag{1}$$

where a, b coefficients calculated as follow

$$b = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{2}$$

$$a = \bar{y} - b\bar{x} \tag{3}$$

where $\bar{x}$ and $\bar{y}$ are the statistical means of x and y coordinates respectively.

Fig. 3 shows a thinned image with LM points and their correspondence fitted line. The slope angle α of the fitted line is then calculated according to Eq.4

$$\alpha = \arctan(b) \tag{4}$$

The second step is to calculate the baseline using the HT, with θ's domain reduced to be $[\alpha \pm 10^{\circ}]$,where $[\pm 10^{\circ}]$ is the empirically observed inaccuracy of LM. We first discretize the θ and ρ parameters and then for each point (x_i, y_i) in the image space we calculate $\acute{\rho}$ as stated in Eq.5:

$$\acute{\rho} = x_i \sin\acute{\theta} + y_i \cos\acute{\theta} \quad \forall\acute{\theta} \in [\alpha - 10^{\circ}, \alpha + 10^{\circ}] \tag{5}$$

Next, each point in the image space will vote for bins that could have generated it in the hough accumulator A, and votes will be accumulated in A according to Eq.6

$$A(\acute{\rho}, \acute{\theta}) = A(\acute{\rho}, \acute{\theta}) + 1 \tag{6}$$

Finally, $\acute{\rho}$ and $\acute{\theta}$ with the maximum number (global maxima) of votes will be considered as the parameters of the word baseline as showed in Eq.7.

$$\arg\max_{\acute{\rho},\acute{\theta}} A(\acute{\rho}, \acute{\theta}) \tag{7}$$

Fig. 4, shows an example of the results, where Fig. 4(A) is the original image, Fig. 4(B) is the skew corrected image and the estimated baseline according to LMR only. Fig. 4(C) shows the result of the skew correction and the estimated baseline of the word using the proposed technique. The reader can clearly see the improvement.

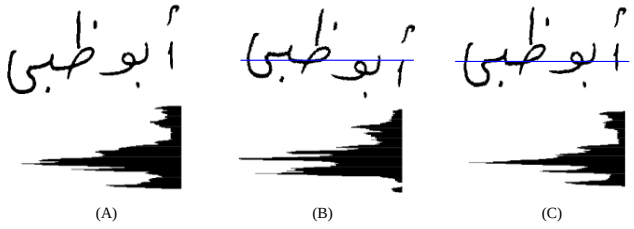
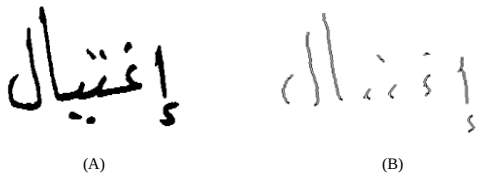


Figure 4. Skew correction and baseline estimation, (A) Original binary image and its corresponding horizontal projection profile, (B) Corrected version using LMR only, (C) Corrected version using the proposed technique.

2) Slant Correction: Slant angle is the angle, which vertical strokes make with the absolute vertical direction. In order to reduce variability within handwritten characters' classes, it is necessary to normalize slant variations [16]. As for the segmentation process, slant correction improves accuracy, since spaces between vertical strokes will be increased.

Since it is being observed, that vertical projection histograms of the slant free images have higher and clear peaks compared to the slanted ones. Thus projection profile based technique calculated upon the horizontal gradient image at various shearing angles in the range $[\pm 45]$, is used for estimation of the slant angle. The reasons behind choosing the horizontal gradient image for calculation are, first, vertical strokes will be emphasized at the expense of horizontal ones, as Fig. 5(B) shows. Second, computation cost will be reduced, since relatively fewer pixels need to be processed.



(A) (B)

Figure 5. Horizontal gradient, (A) The binary word image, (B) The horizontal gradient counterpart.

Given (x, y) the coordinates of pixel in an image i, their sheared counterparts $(\acute{x}, \acute{y})$ in the sheared image $\acute{i}$ are calculated according to Eq.8

$$\acute{x} = x - y.\tan(\alpha), \quad \acute{y} = y \tag{8}$$

where $\alpha \in [\pm 45]$ is the shearing angle.

For each sheared image we calculate vertical histogram H as stated in Eq.9.

$$H(\acute{x}_L; \alpha) = \sum_{k=0}^{\infty} \acute{i}(\acute{x}_L, \acute{y}_k) \tag{9}$$

Then the variation for every two consecutive profiles is calculated as in Eq.10.

$$V(\alpha) = \sum_{L=0}^{\infty} [H(\acute{x}_L; \alpha) - H(\acute{x}_{L+1}; \alpha)]^2 \tag{10}$$

And the sheared angle $\acute{\alpha}$ is calculated as the angle associated with maximum variation, according to Eq.11.

$$\acute{\alpha} = \arg\max_{\alpha} V(\alpha) \tag{11}$$

Fig. 6(A) shows a slanted word image and its corresponding vertical histogram, and Fig. 6(B) shows its slanted free version and its vertical histogram.

(A) (B)

Figure 6. Slant correction, (A) A binary word image and its correspondence vertical projection profile, (B) The Slant corrected version and its correspondence vertical projection profile.

B. Characters Segmentation

As mentioned above our segmentation approach conducted in two steps. In the first step, a careful analysis of the x-axis coordinates of the connected components is performed. The result is words images with resolved subword overlapping. The second step is to perform topological feature based segmentation for characters' representatives.

Before detailing our approach, it is helpful to start by recalling some definitions that are thought to be necessary for the clarity of subsequent definitions and notations.

Firstly, let P refers to any foreground pixel in the thinned word image $g(x, y)$, and let $N_8(P)$ denotes the 8-neighborhood set of P. Secondly, by examining each $P \in g(x, y)$ a set of feature points are identified, which we call **C**ritical **F**eature **P**oint (CFP). CFP set contains further four subset that are listed below:

i. The first subset is **E**nd **P**oints (EP) depicted in Blue in Fig. 7, which are all pixels with only one pixel in its 8-neighborhood set.
$$EP = \{P | N_8(P) = 1\} \tag{12}$$

ii. The second subset is Branch Points (BP) depicted in bright Green in Fig. 7, which are all pixels where its 8-neighborhood set contains only 3 or 4 pixels.
$$BP = \{P | N_8(P) = 3 \vee P | N_8(P) = 4\} \tag{13}$$

iii. The third subset is Dot Points (DP), which is the union of the set of all isolated pixels, and the set of pixels that belong to connected components (CC) that are less in size than an adaptive threshold T proportional to the estimated character size calculated upon the thinned text image. Depicted in Cyan in Fig. 7
$$\begin{aligned} DP = &\{P | N_8(P) = 0\} \\ &\cup \{P | P \in CC \wedge size(CC) < T\} \end{aligned} \tag{14}$$
where $T \leq \mu$, and μ is the mean of the area of all CC, that are not intersecting the baseline.

iv. The fourth and last subset is the Loop Points (LP), which are all On remained pixels of the thinned text image after performing the $flood-fill$ (ff) algorithm on it. depicted in Red in Fig. 7
$$LP = \{P | P \in ff(i(x, y))\} \tag{15}$$

Given the aforementioned four subsets the CFP set can be defined as the union of all the four subsets. Fig. 7 shows a thinned text image with all possible CFPs, that will be utilized later to guide the characters' segmentation process.

$$CFP = \cup\{LP, EP, BP, DP\} \tag{16}$$

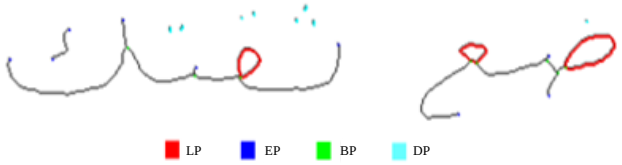


Figure 7. CFP Feature Points, thinned text image with all possible CFPs.

Next, we present the first step of the proposed segmentation approach.

1) Resolving of Sub-words' Overlapping: For resolving the sub-words overlapping, we first find the word baseline as stated above. Then upon finding the baseline, we differentiate between two types of connected components (CC). The first is what we call main (CC)s, which are all (CC)s that intersecting with the baseline's "y" coordinate. The second are what we call auxiliary (CC)s, which are all CCs that are not intersecting the baseline's "y" coordinate. Fig. 8 shows an example of word images, where the main CCs are (1,2,4, and 6), and the auxiliaries are (3,5,7, and 8) and the horizontal blue line representing the baseline.

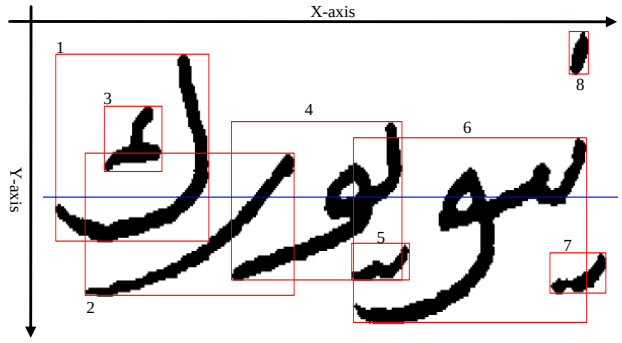


Figure 8. Sub-words overlapping example.

After identifying the main CCs, we conduct a distance analysis on their bounding boxes along the x-axis, in order to identify the baseline overlapped main CCs and their correspondence overlapping distances. In Fig. 8 for example, main CCs that are overlapping are (1,2), (2,4), and (4,6). Another distance analysis is performed against the auxiliary (CC)s, so each can be assigned to its correspondence main CC according to the following rules:

i. If an auxiliary CC is overlapping only a given main CC along the x-axis, then assign the auxiliary to the main. So in Fig. 8 for example, auxiliaries number (8, 7) will be assigned to the main CC number "6".
ii. If an auxiliary that is above the baseline, is completely contained in the bounding box of a main CC, then assign the auxiliary to the main regardless of any main CC that may overlap it along the x-axis. So "3" will be assigned to "1" in Fig. 8.
iii. if two main CC are overlapping an auxiliary under baseline, like in case of "5" that is overlapped both "4" and "6", we calculate the absolute distance along the y-axis, between lower bounding box of the auxiliary, and the lower bounding box of the overlapping main CCs; the one with minimal distance wins the auxiliary. So "4" wins "5" in Fig. 8 for example.
iv. In case auxiliary is above the baseline and overlapping multiple main CCs, the absolute distance along the y-axis is measured between its lower bounding box "y" coordinate and the upper "y" coordinates of the overlapping main CCs bounding box; the main CC with the minimum distance wins.

Even though the aforementioned rules solve for almost all the cases, there are some extreme cases where auxiliaries are not overlapping any main CC. In these cases, auxiliary is assigned to the direct next main CC on the left [1]. After assigning the auxiliaries to their corresponding main CCs, we computed the sub-words borders along the x-axis against all its elements (auxiliary CCs and main CCs). The sub-word's bounding box left border, is computed to be the farthest left border among all sub-word elements. Likewise, the right border is selected to be the farthest border to the right.

Eventually, a final distance analysis is preformed against the new sub-word borders and the overlapping solved by shifting away the overlapped sub-words. Fig. 9 shows two examples (A) and (B) and their correspondence sub-word overlapping free version. As a result of this pre-segmentation step, sub-words are separated by empty columns that make their segmentation a straight forward process.

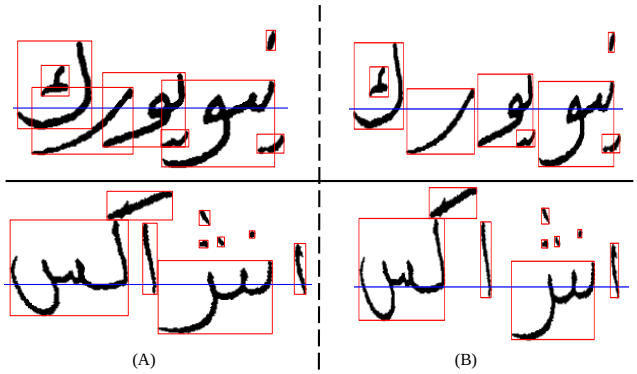


Figure 9. Resolving sub-word's overlapping, (A) Overlapped sub-words versions, (B) Results of overlap resolving procedure.

2) Segmentation of Character Representative : Providing sub-word overlapping is solved, we turn to the issue of segmentation of sub-words into their constituent characters' representatives. Given that Arabic characters have their boundaries in columns with the minimum number of pixels (only one pixel in the thinned version), our segmentation approach starts by generating a set C of columns' indices as candidates for segmentation, where the elements of C are all column indices within the thinned image $g(x, y)$, containing only one foreground pixel. We developed the segmentation algorithm presented in [7], in a way that we use broader set of segmentation candidates instead of using

[1]This is due to the fact that Arabic text is written from right to left, and writers usually writing main CC first, then auxiliaries. As a result auxiliaries are appearing shifted to the right away from their correspondence sub-words.

the set of only the contour's local minima. We observed that using local minima as candidates for segmentation is risking good segmentation, because local minima often occur inside many of Arabic characters main strokes. So we decided to generate a large set of segmentation candidates, and we did not restrict our candidates to be only local minima.

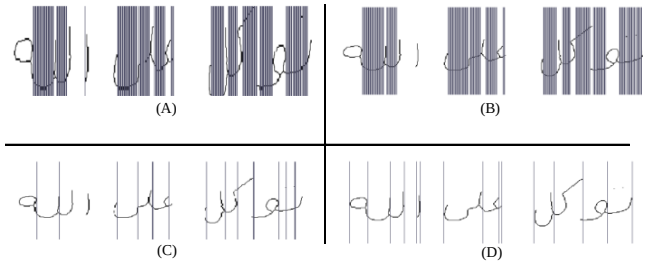


Figure 10. Cut candidates election, (A) Depicts all possible cut candidates. (B) Candidates after excluding column that contains EP, LP, BP or DP. (C) Candidates after excluding candidates with direct left neighbor. (D) The result after applying the proposed heuristic rules

Fig. 10(A) is depicting the columns' candidates for segmentation of the text image, the reader can notice that each column containing only one pixel is elected as a candidate. The next step is to exclude from the candidates set all columns that are intersecting with any CFP, this is to say that LP, EP, BP and DP columns cannot be in the same time a cut point, otherwise we lose important character features. Fig. 10(B) shows a text image after excluding columns' candidates that are intersecting CFP columns. The reader can notice in Fig. 10(B) that very few candidates are excluded comparing to Fig. 10(A), this is because, it is quit seldom that an LP or a BP column contains only one pixel, so it will not be chosen as a candidate in the first place.

The next step is to scan the list of candidates, starting from the most right one to the left, electing the left neighbor from each two adjacent segmentation candidates. Fig. 10(C), shows the result, we can easily notice the significant reduction in the number of candidates for segmentation. The candidates set obtained so far is an important improvement over the previous candidate sets. However, in order to resolve issues like, for example the over-segmentation in Fig. 10(C) (*the letter ت (TAA), the first letter from the right is over-segmented into three parts*). We formulate four conditions to increase segmentation accuracy.

To ease notation of conditions, we will write m a subscript to CFP or/and $CFPs$ elements, to refer to the respective column index. Also, we will use c_i , c_j to refer to any two column indices in the thinned image $g(x, y)$, that are chosen to be candidates. The finale election process is performed by applying the following condition on the candidates:

i. First condition, is saying that if there are two consecutive cut candidates and there is no CFP in between then delete from the list the one on the right, this condition can be formulated as following;

$$\forall\{c_i, c_j\} \in \{C\}|c_i > c_j, \\ if\ \{CFP_m\} \notin [c_i, c_j] \Rightarrow c_i \notin C \tag{17}$$

ii. Second condition, if the direct neighboring on the left is a column contain pixel of DP, then delete the candidate from the list. This can be notated as following:

$$\forall c_i \in C\ if\ \exists(c_i + 1) \in DP_m \Rightarrow c_i \notin C \tag{18}$$

where DP_m, is the set of columns contain DP pixels.

iii. Third condition saying that if there is a branch point column BP or Loop point column LP before encountering another candidate then we elect the candidate as cut point, the notation version of the condition is:

$$\forall\{c_i, c_j\} \in \{C\}|c_i > c_j, \\ if\ \exists(BP_m \vee LP_m) \in [c_j, c_i] \Rightarrow c_i \in C \tag{19}$$

iv. If the next column contains an end point EP_1, which is in the same time not an end of stroke, then flow the contour starting from EP_1 down to the left, if another end point EP_2 is encountered (before BP or LP) which, not on the contour and is an end of the stroke, then elect the candidate as a cut point.

Finally, we insert a cut candidate direct before and after every connected component.

Fig. 10(D) illustrates the final segmentation results, and Fig. 11, shows zoomed in segmentation's result of an Arabic sentence.

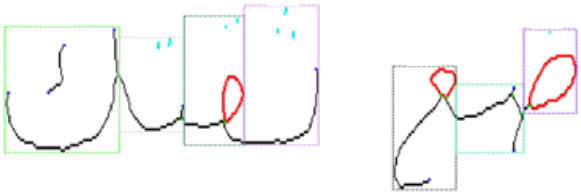


Figure 11. Characters segmentation, characters' representatives are bounded by rectangles.

III. Experimental Results

We experimented our proposed methodology on an under construction database of handwritten Arabic words, that contains more than 3000 Arabic words images, collected from more than 30 persons. Our results are very satisfactory, and to our knowledge outperforming literature available results so far. We have tested a system implemented according to the proposed approach on a set of 200 different words' images. Fig. 12, illustrates some of the results, where complete success is reported in 72% of cases, this means the system accurately discovers the character representatives' borders. Fig. 12(A), Fig. 12(B), Fig. 12(C), Fig. 12(D) illustrates examples of 100% success segmentation of character representatives, where each is bounded in a rectangle. Partial

success is reported in 28% of cases, we have empirically noticed that 9% of such cases are generated when CFPs occur inside the character instead of on its borders, leading to what we call an over segmentation, where a part of stroke is regarded as a character representative, which, in fact, it is not. This problem is specific to characters س and ش (SIEN and SHIEN), and Fig. 12(A) illustrates an example, the black arrow is pointing out to where it occurs. The other 19% of cases happen when CFPs cease to exist between two consecutive characters leading to segment them as a representative for one character. Fig. 12(F), shows such problem, where its position indicated with the arrow. This problem is called under-segmentation, and it is specific for cases, when the second character to left is connected ا (ALF) or connected ل (LAM) with sheared distortion angle to the left. We think that those problems can be solved, either by expanding the CFP set to contain more features points like Local minima points for example and then accordingly modify and adding heuristic rules, or they can be solved in subsequent recognition phases like in the post-processing phase for instance, where the recognition results can be corrected against lexicons using different text retrieval techniques.

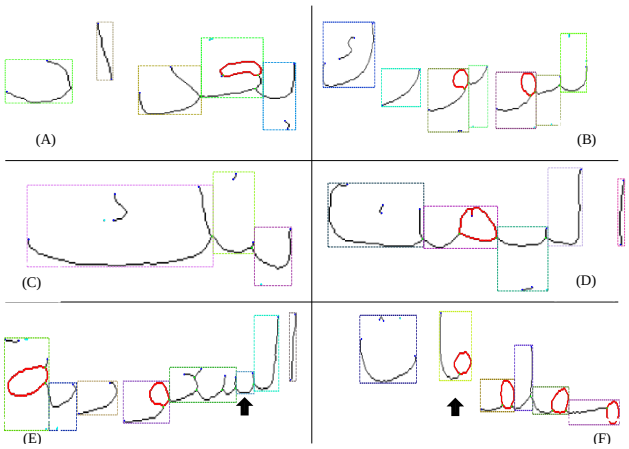


Figure 12. Characters segmentation, (A)-(D) Successful segmentation of characters representatives, (E)-(F) Partial success of segmentation.

IV. Conclusion and Future Works

In this paper, we proposed dual-phase segmentation approach that starts first by sub-word borders' identification and resolving overlapping among them, and then topological features based segmentation is taking place according to a set of heuristic rules. The dual-phase segmentation is preceded by an extensive handwriting specific pre-processing phase in which issues like morphological enhancement, skew correction, and slant correction are dealt with. Even though results were quite satisfactory, future works may investigate issues like, expanding CFP set by adding more topological features and correspondingly more heuristics. Moreover, a cyclic segmentation-recognition based approach is expected to improve results further. In such approach character representatives segmentation proved against the results of the recognition phase. As an in-between approach, of segmentation based approaches with their relatively high error-prone tendency and holistic based approach with their small and restricted domains [17]. A holistic sub-words based approach can be another alternative to attack the problem of off-line handwritten Arabic text recognition.

References

[1] L. M. Lorigo and V. Govindaraju. "Offline Arabic handwriting recognition", *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, Vol. 28, No. 5, pp.712724, May 2006.

[2] M. Elzobi, A. Al-Hamadi, L. Dinges, B. Michaelis. "A Structural Features Based Segmentation for Off-line Handwritten Arabic Text", International Symposium on Image/Video Communications over fixed and mobile networks (ISIVC2010). Rabat, Morocco, Sep.30-Oct.02, 2010.

[3] I. Abuhaiba, M. Holt and S. Datta. "Recognition of Off-Line Cursive Handwriting", Computer Vision and Image Understanding, Vol. 71, No. 1, pp. 19-38, July 1998.

[4] N. A. Shaikh, Z. A. Shaikh, and G. Ali. " Segmentation of Arabic Text into Character for Recognition", IMTIC 2008, CCIS 20, Springer Berlin/Heidelberg, pp.11-18, 2008.

[5] H. Almuallim, S. Yamaguchi. " A Method of Recognition of Arabic Cursive Handwriting", *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, Vol. 9, No. 5, pp.715-722, Sep. 1987.

[6] Z. Al Aghbari, S. Brook. "HAH manuscripts: Aholistic paradigm for classifying and retrieving historical Arabic handwritten documents", Journal of Expert Systems with Applications , Vol. 36, Nr. 8, (2009), pp. 1094210951.

[7] A. A. Atici and F. T. Yarman. "A Heuristic Algorithm for Optical Character Recognition of Arabic Script", Signal Processing., Vol. 62, No. 1, pp.87-99, Oct. 1997.

[8] X. Ding, and H. Liu. "Segmentation-Driven Offline Handwritten Chinese and Arabic Script Recognition", Springer Berlin/Heidelberg, LNCS 4768, pp.196-217, 2008.

[9] D. Motawa, A. Amin, and R. Sabourin. " Segmentation of Arabic Cursive Script", ICDAR, pp.625-628, 1997.

[10] E. Arias-Castro and D.L. Donoho. "Does median filtering truly preserve edges better than linear filtering?", Annals of Statistics, Vol. 37, No. 3, pp.11721206, 2009.

[11] L. Lam, S. Lee, and C. Suen. "Thinning Methodologies - A Comprehensive Survey", *Pattern Analysis and Machine Intelligence, IEEE Transaction on*, Vol. 14, No. 9, pp.869-885, Sep. 1992.

[12] Z. Razak et al. "Off-line Handwriting Text Line Segmentation: A Review", IJCSNS, Vol. 8, No.7, Jul. 2008.

[13] V. Beusekom, F. Shafait and T. M. Breuel. "Combined orientation and skew detection using geometric text-line modeling", IJDAR, Vol.13, No.2, pp.79-92, Springer-Verlag Berlin, Heidelberg, Jun. 2010.

[14] T. M. HA and H. Bunke. "Image processing methods for document image analysis," *in Handbook of Charcter Recognition and Document Image Analysis*, Singapore:World Scientific Publishing Co. Pte. Ltd.,1997, pp.1-47.

[15] M. Wienecke. ”Videobasierte handschrifterkennung”, Ph.D. dissertation, Bielefeld university, Bielefeld, Germany, 2003.

[16] N. Arica and F. T. Yarman-Vural.”An Overview of Character Recognition Focused on Off-Line Handwriting”, *Systems, Man and CyberneticsPart C: Applications and reviews, IEEE Transaction on*,Vol. 31, No. 2, pp.216-233, May 2001.

[17] V. Lavrenko, T. M. Rath and R. Manmatha. ”Holistic Word Recognition for Handwritten Historical Documents”, DIAL04, pp.278-287, 2004.

Chroma Reconstruction from Inaccurate Measurements

Alexander Balinsky
Cardiff University, UK
BalinskyA@cardiff.ac.uk

Nassir Mohammad
Cardiff University & HP Labs, UK
MohammadN3@cardiff.ac.uk

ABSTRACT

Non-linear filter responses of natural colour images have been shown to display non-Gaussian heavy tailed distributions which we call sparse. These filters operate in the YUV colour space on the chroma channel U (and V) using weighting functions obtained from the gray image Y. In this paper we utilise this knowledge for denoising the chroma channels of a colour image from inaccurate measurements. In our model the U (and V) elements are affected by noise, with a good version of the gray image Y obtainable through existing methods. We show that accurate reconstruction of the chroma components can be accomplished by solving an L1 constrained optimisation problem, where the sparse filter response on natural images is used as a regularization term. This scheme gives comparable results to leading commercial and state of the art denoising algorithms, and exceeds for chroma noise that does not correlate with the luminance structure.

Keywords: Natural images, filter response, sparse distributions, denoising, L^1 optimisation.

1 INTRODUCTION

Denoising is a fundamental problem in image processing due to the fact that images, no matter their content, usually contain some degree of noise. This is often regarded as a form of image degradation and the goal of denoising algorithms are to form an estimate x' of the the original image x given the observed noisy version x^*, modeled as

$$x^* = x + n, \tag{1}$$

where n is the matrix of the random noise pattern.

The principal causes of noise in digital images arise during image acquisition (digitization) and/or transmission. This can be caused by several factors such as low light levels, sensor temperature, electrical interference, malfunctioning pixels and interference in the channels used for transmission. The distribution of noise can be several, such as white, impulse or multiplicative, each giving its own characteristic form of degradation.

Various algorithms have been introduced with success over the past few decades for denoising images. The proposals, in their original form, have sparked an abundant literature resulting in many improvements in quality and speed. These algorithms can be categorized into several groups including Wavelets, Bilateral filtering, Anisotropic diffusion, Total variation and Non-local methods. Readers are advised to see [BUA05] and [MAI08] for comprehensive reviews and comparisons of the best available versions together with powerful novel approaches.

Some recent algorithms to mention include [LIU08] where the authors propose a unified framework for two tasks: automatic estimation and removal of colour noise from a single image using piecewise smooth image models. Their segmentation-based denoising algorithm is claimed to outperform current methods. This paper also contains an interesting introduction that discusses the current state of the art methods for image denoising. Another recent algorithm which claims to lead to excellent results is C-BM3D [DAB07]. In this scheme the authors propose an effective colour image denoising method that exploits filtering in a highly sparse local 3D transform domain in each channel of a luminance-chrominance colour space. For each image block in each channel, a 3D array is formed by stacking together blocks similar to it. The high similarity between grouped blocks in each 3D array enables a highly sparse representation of the true signal in a 3D transform domain, thus a subsequent shrinkage of the transform spectra results in effective noise attenuation.

The importance of denosing in image processing has also led to many commercial and freely available software. These include Neat Image, Noise Ninja, DenoiseMyImage, Photoshop, Topaz Denoise, Gimp and many more. The programs often incorporate a host of image enhancement tools to collectively remove typical forms of image degradation. A full evaluation of so many programs is difficult, especially since each has parameters which a user can change for subjective suitability. However, from general usage and reading it has been found that Noise Ninja and Neat Image are among the best used noise reduction programs. DenoiseMyImage is also a current alternative that uses a modified form of the state of art non-local means method. Readers may view [ALM] for a comprehensive *user* comparison of current software.

Denoising algorithms are usually fed a noisy *RGB* image corrupted in each channel. Most methods have

been formulated as a channel by channel or vectorial model. In the former case the *RGB* values are mapped to a colour space such as YUV or *Lab* or any other suitable space to separate the luminance and chroma, with the denoising algorithm *usually* applied to each band. Since the luminance channel contains the main structural information and chroma noise is more objectionable to human vision (as opposed to the film grain appearance of luminance noise), separation allows more intensive denoising of the chroma channels without too much loss of detail. These models take into account the human perception of colour and allow us to handle the particular characteristics of the noise affecting each component. Methods based on their luminance-chromatic decomposition are well known for their excellent results with [DAB07] being a recent example. Furthermore, in the process of transmission, the reduction of bandwidth for the chroma allows errors and artifacts to be more easily compensated for than using a typical *RGB* model.

In this paper we propose a novel algorithm for removing noise from real images and also white and impulse noise from the chroma channels of an image in the colour space YUV, where a *good* version of the Y component is obtainable. (Due to the similarity of the colour components, from here on we interchangeably mention either the U or V channel, where analysis of the other is obtained by substitution). Algorithms such as those in [DAB07], [FOI07] and [BOR05] have successfully exploited the information in the luminance channel for effectively filtering the chroma components. In line with this philosophy our approach utilises the non-linear filter response distributions observed in [BAL09] as a regularization term (a prior, in Bayesian analysis) to penalize solutions that don't give a *desired* sparse solution when filtering.

The rest of the paper is laid out as follows: section 2 describes the motivating details behind our regularization function. Section 3 outlines our denoising procedure while section 4 gives results for denoising images. Section 5 summarizes the paper and directions for future work.

2 REGULARIZATION USING THE SPARSE DISTRIBUTION OF THE FILTER RESPONSE

Our approach in denoising the chroma components involves introducing a regularization term which incorporates knowledge of the statistics of natural images. More specifically we consider the recent non-linear filter response distributions of natural images observed in [BAL09]. In that paper the authors show that colour images, when filtered by the following:

$$F(U)(\mathbf{r}) = U(\mathbf{r}) - \sum_{\mathbf{s}\in N(\mathbf{r})} w(Y)_{\mathbf{rs}} U(\mathbf{s}), \qquad (2)$$

display non-Gaussian heavy tailed distributions, i.e. sparse. Here $\mathbf{r}$ represents a two dimensional point, $N(\mathbf{r})$ a neighborhood (e.g. $3x3$ window) of points around $\mathbf{r}$, and $w(Y)\mathbf{rs}$ a weighting function. The proposed filter thus takes a point $\mathbf{r}$ in U (or in V) and subtracts a weighted average of chroma values in the neighborhood of $\mathbf{r}$. The $w(Y)\mathbf{rs}$ is a weighting function that sums to one over $\mathbf{s}$, large when $Y(\mathbf{r})$ is similar to $Y(\mathbf{s})$, and small when the two intensities are different. (See [BAL09] for further details).

The response of the filter can be modeled by a generalized Gaussian distribution (GGD)

$$f(x) = \frac{1}{Z} e^{-|x/c|^{\alpha}}, \qquad (3)$$

where Z is a normalising constant so that the integral of $f(x)$ is 1, c the scale parameter and α the shape parameter. It is found for natural images that $\alpha < 1$ which results in a non-convex function. However, due to the recent success of L^1 optimization in recovering approximately sparse signals [CAN06], we convexify our model i.e. take $\alpha = 1$, and use this as a regularizer in (4).

3 CHROMA DENOISING PROCEDURE

We consider real noisy *RGB* images that have been corrupted by unknown noise which are then transformed to the YUV colour space. Due to the properties of the underlying natural colour images, such as high correlation between R, G, and B channels, we note that Y has higher SNR than U and V and that it contains most of the valuable information such as edges, shades, objects, texture patterns, etc. The U and V contain mostly low-frequency information with iso-luminant regions, i.e. variation in only U and V, being unlikely. Thus removing chroma noise through knowledge of gray information is plausible. We chose to use Neat Image or DenoiseMyImage when appropriate to denoise the Y channel when needed. We additionally used them as a benchmark for testing our algorithm. Furthermore, our algorithm is also tested against images in the YUV space suffering from impulse noise only in the chroma channels.

Thus, given the noisy chroma component U^* and a denoised gray image Y, our task is to recover a good approximation U' of the original element U. This model results in the following optimisation scheme,

$$argmin_{U'} \; ||F \cdot U'||_1 + \lambda ||U' - U^*||_d. \qquad (4)$$

Given an $n \times m$ image, (we abuse the notation a little and have) F here is an $nm \times nm$ matrix whose rows correspond to filtering a single pixel where U' and U^* are $nm \times 1$ column first rasterized vectors. U' is the estimate we seek of U, while U^* is the noisy observation of U.

The first term is our penalizing function which takes small values for desirable solutions and the second is the *fidelity* term. The parameter d is taken to be either 2 or 1 reflecting the norms proposed in the measurement of the distance between the two vectors. In words, this optimisation scheme searches for the estimate image U' with the sparsest filter response and with the second term encouraging the solution to be close to a noisy chroma measurement U^*.

For an image assumed to be corrupted by Gaussian noise our reconstruction process involves solving (4) with $d = 2$, where the fidelity term encourages solutions to be close to the noisy version in the L^2 sense. When the noise is taken to be impulsive and affecting the image at random points by taking extrema values, we solve (4) with $d = 1$. Modifying the fidelity term to $d = 1$ (i.e. L^1 norm) has been studied with success within the Total Variation framework, as reviewed in [CHA05].

An important parameter in our algorithm is the value of λ which controls the relative weight of the difference between the noisy channel and the solution. Too small a value and the optimisation results in an overly smoothed output, while too high a value results in a solution that is too close to its noisy version. We found experimentally that $\lambda \in (0,5]$ gave the best results, with half-integer increments for optimality.

4 RESULTS

Our optimisation problem was solved using CVX [GRA] which is a convex programming package implemented in Matlab. The images that we used are of sizes in the region of 200×200 pixels, which took on average a couple of minutes to denoise. However, our aim here is not to pose a fast algorithm but only to show the applicability of such a scheme for denoising chroma channels. The algorithm is parameterised by the value of λ whose value is given in the text accompanying the figures.

Fig. 1(a) shows an example *RGB* image which is made severely noisy by adding Gaussian noise of mean zero and variance 0.01 to all the channels as shown in (b). (c) shows the denoised image obtained using Neat Image and (d) the result obtained using DenoiseMyImage. Neat Image was used at maximum setting while DenoiseMyImage was used at an adjusted medium level to obtain the best results. Neat image still left considerable noise like artifacts in the image, while DenoiseMyImage gave a less noisy but much smoother output. The result using our algorithm is shown in (e) where we used DenoiseMyImage to denoise the gray component. Visually comparing the results shows that our algorithm gives an intermediate result which is better than using NeatImage, while the colours are much more vibrant and appear sharper than when using DenoiseMyImage. This is also further justified by the peak signal to noise ratios (PSNR) which quantify the results, and shows our algorithm having a higher but similar value.

The next examples focus on real world images where the type of noise affecting the image is unknown. We begin with Fig. 2(a) which shows an image that is severely affected by colour noise. This is typical of an image taken in low light conditions with high ISO settings. (b) shows the image having been denoised using Neat Image. This program requires a suitable region to be selected for noise estimation, after which luminance and chrominance noise reduction can be individually adjusted. We required 100% noise reduction on all components due to the high amount of noise present in the image. (c) shows our algorithm where the luminance channel was denoised using Neat Image and the filter matrix F constructed from it for reconstructing the chroma channels. (d) shows the result of using DenoiseMyImage. We observe that our algorithm gives similar noise reduction compared to the existing methods, although on close inspection our result gives less colour aberrations.

Fig. 3(a) has been taken from some examples given on the Neat Image website. This is a crop of a television frame captured with a computer TV card. The image has strong colour banding visible across all the image caused by the electric interference in the computer circuitry. Similar banding is sometimes observed in digital camera images (caused by interference too). The banding degradation does not affect the luminance, however all channels still show grain like noise. (b) shows the best Neat Image result obtainable by denoising the chroma and luminance at 100%. However, the banding is still evident in the result. (c) is the result of our algorithm which clearly removes the noise. (d) is the best result obtainable using DenoiseMyImage which is still unable to remove the banding noise.

Our algorithm is able to remove this type of noise by filtering only the chroma channels and using Neat Image for clearing the fine grain luminance noise. The result is free of the colour banding and (f) shows that the V channel does not display any of this degradation against the V channel when using Neat Image (e). We are able to attain this result as we are filtering the chroma channels through taking account of the underlying gray level structure. Since the colour banding is not appearing in the luminance, minimisation of the filter response favours areas of homogeneous colours while the fidelity term bounds the colours to being close to the original.

The final two examples illustrate the flexibility of the model in handling chroma noise taking a different distribution. Fig. 4 shows an example of a clean image (a) which is transformed to the YUV colour space and impulse noise of density 0.05 added to the U and V channels only. Our algorithm, with the fidelity term

measuring L^1 norm, is able to denoise such that the recombined *RGB* image shown in (b) is visually identical to the original. The detailed look at the chroma components reveals no sign of the impulse noise, while the PSNR is of a good value.

Fig. 5 shows another example of an image that has been corrupted by impulse noise and reconstructed. (a) shows the original image, (b) the *RGB* image with noise having been added to only the chroma channels and (c) shows our reconstructed image. The results illustrate again that noise has been successfully removed to a very high standard with good PSNR values, and this is further justified by looking at the chroma channels which have had their impulse noise removed. Neat Image and DenoiseMyImage are unable to effectively denoise the images affected by impulse noise. Instead we obtain a 'washed out' look with the impulse points still remaining. An example is shown by (d).

5 CONCLUSION

We have illustrated how knowledge of the statistics of natural images can be incorporated into an effective denoising scheme. Our objective was to propose a novel algorithm for removing chroma noise from digital images by operating in a luminance-chrominance colour space. We utilised the sparse filter response distribution of the filter studied in [BAL09] as a regularization term, and introduced a quadratic fidelity term to ensure the solution remained close to the original. This model allowed us to denoise real images with results comparable to current alternatives. The flexibility of the model was also shown by its ability to handle chroma impulse noise very effectively, giving results that are virtually identical to the original image. This was accomplished by altering the fidelity term to measure L^1 norm and shows concentration on gray level denoising gives sufficient information for colour channel reconstruction.

In future it would be most useful to robustly test this approach across diverse datasets of images and also in other colour spaces where we may observe increased performance. We are also looking at algorithms for solving the optimisation scheme much more quickly and looking at applying the approach to denoising hyperspectral images.

ACKNOWLEDGMENTS

This work was supported in part by grants from EPSRC and Hewlett Packard Labs awarded through the Smith Institute Knowledge Transfer Network.

(a) (b) (c) (d) (e)

Figure 1: Denoising example. (a) shows the original image, (b) the image with Gaussian noise added to all *RGB* channels. (c) is the result using Neat Image at maximum filtering. (d) shows the denoising result using DenoiseMyImage. (e) is the result obtained using our algorithm. PSNR: (c) 26.69, (d) 26.35, (e) 27.20 ($\lambda = 5$)

REFERENCES

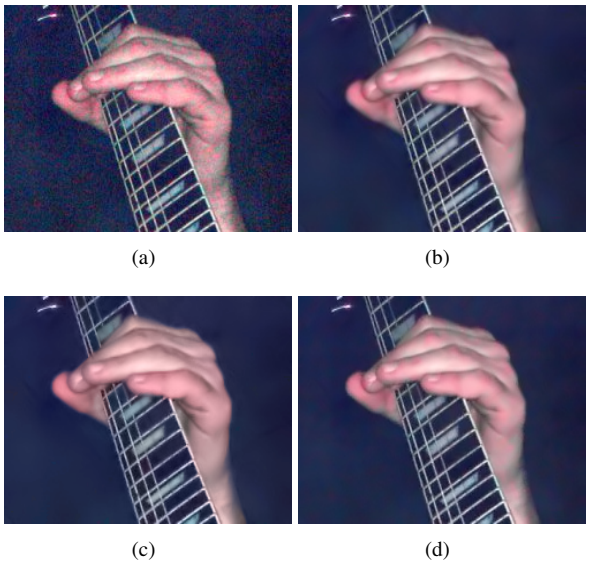
[BAL09] A. Balinsky and N. Mohammad, "Non-linear filter response distributions of natural colour images." *LNCS* 5646, pp. 101-108. *Springer-Verlag Berlin Heidelberg* 2009.

[BUA05] A. Buades, B. Coll, and J. M. Morel, "A review of image denoising algorithms, with a new one, Multiscale Modeling & Simulation, vol. 4, no. 2, pp. 490530, 2005.

[MAI08] J. Mairal, M. Elad, and G. Sapiro, "Sparse representation for color image restoration", IEEE Transactions of image processing, vol. 17, No. 1, January 2008.

[LIU08] C. Liu, R. Szeliski, S.B. Kang, C.L. Zitnick, W.T. Freeman, "Automatic Estimation and Removal of Noise from a Single Image", IEEE Transactions on pattern analysis and machine intelligence, vol. 30, NO. 2, Feb 2008

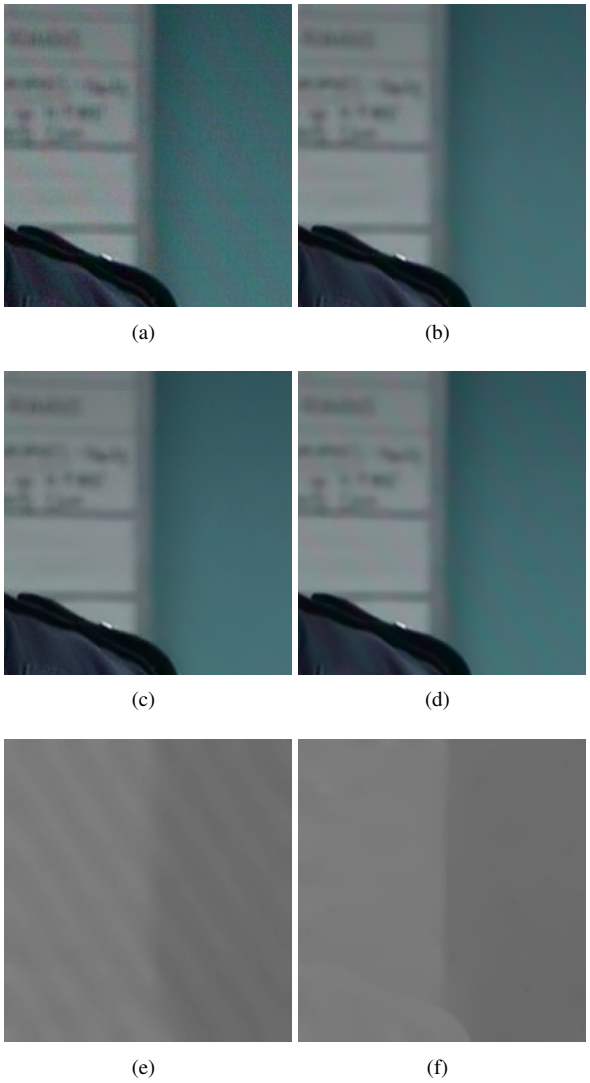
[DAB07] K. Dabov, A. Foi, V. Katkovnik, and K. Egiazarian, "Color image denoising via sparse 3d collaborative filtering with grouping constraint in luminance-chrominance space", ICIP 2007. (Matlab code available at www.cs.tut.fi/ foi/GCF-



(a) (b)

(c) (d)

Figure 2: Real image denoising example. (a) is an image that has been affected by severe chroma noise resulting in the appearance of 'blotches' of colour. (b) shows the denoised image obtained using Neat Image and (c) is obtained using our algorithm. (d) is the result obtained using DenoiseMyImage. We observe that all the reconstructions are visually similar, although on close inspection our result gives less colour aberrations. ($\lambda = 0.5$)



(a) (b)

(c) (d)

(e) (f)

Figure 3: Real image denoising example. (a) shows an example image affected by chroma noise that appears as bands in the colour channels. (b) is the result obtained using Neat Image which still leaves evident colour banding. (c) is our result which is able to remove the noise leaving a clean image as the colour banding does not correlate with the luminance structure. (d) is the best result obtained using DenoiseMyImage. (e) shows the banding still remaining in the V channel of the image when using Neat Image, while (f) clearly shows that the banding structure has been removed in our reconstructed V channel. ($\lambda = 0.1$)

BM3D).

[ALM] M. Almond, http://www.michaelalmond.com/Articles/noise.htm

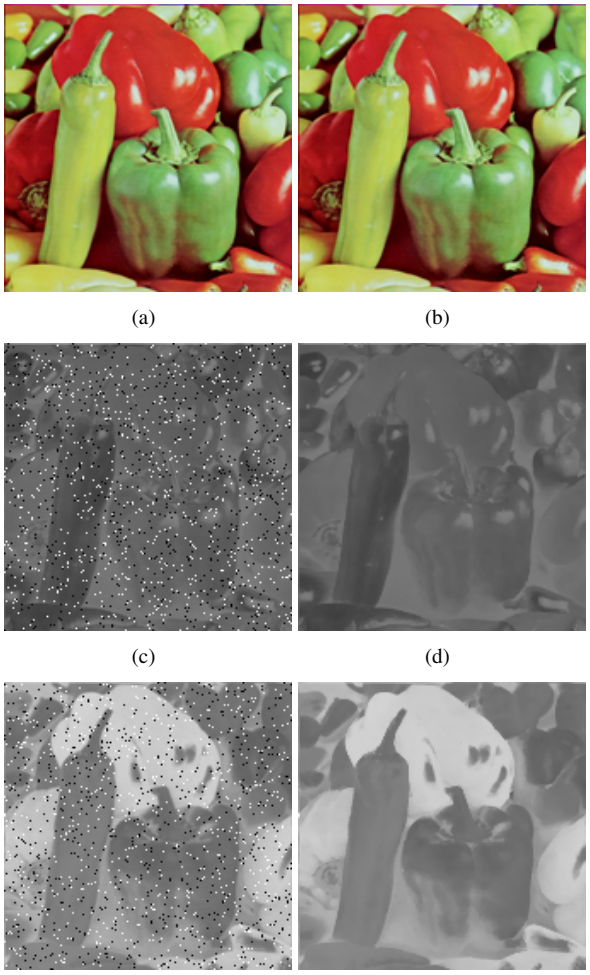
[FOI07] A. Foi, V. Katkovnik, and V. Egiazarian, "Pointwise Shape-Adaptive DCT for High-Quality Denoising and Deblocking of Grayscale and Color Images", IEEE Trans. Image Process., vol. 16, no. 5, May 2007.

[BOR05] D Borkowski, "Chromaticity Denoising using Solution to the Skorokhod Problem", Image Processing Based on Partial Differential Equations, Proceedings of the International Conference on PDE-Based Image Processing and Related Inverse Problems, CMA, Oslo, August 812, 2005.

[CAN06] E. Candes, "Compressive Sampling," *Int. Congress of Mathematics,* 3, pp. 1433-1452,Madrid, Spain, 2006.

[CHA05] T. F. Chan and S. Esedoglu, "Aspects of total variation regularized L^1 function approximation", SIAM J. Appl. Math., 65 (2005), pp. 18171837.

[GRA] M. Grant and S. Boyd, http://cvxr.com/cvx/

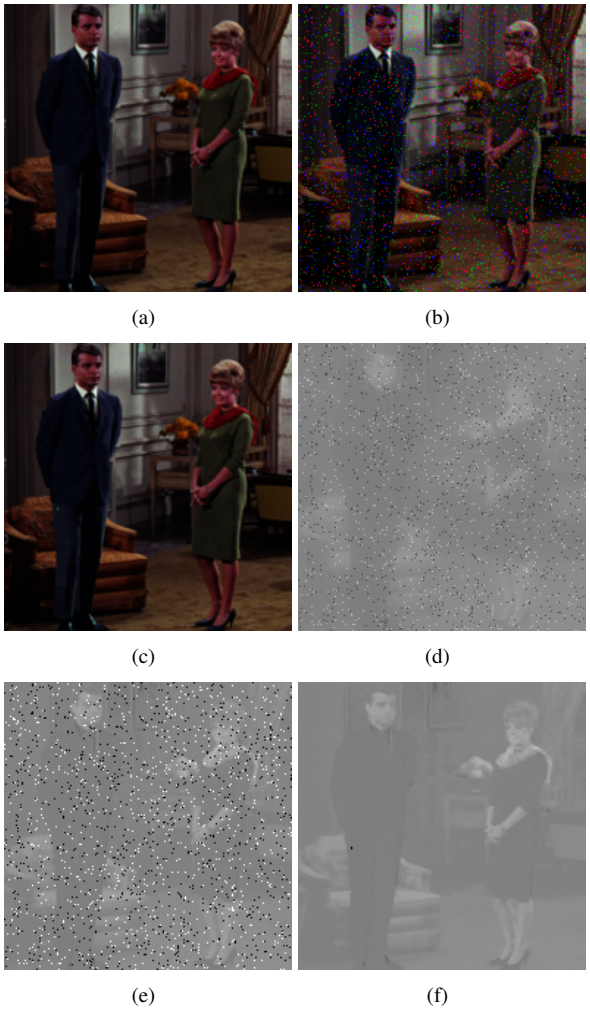


(a) (b)

(c) (d)

(e) (f)

Figure 4: Impulse noise removal example. (a) shows the original image and (c) and (e) illustrate the colour channels with impulse noise added. (b) is the reconstructed image which does not display the impulse noise and is visually identical to the original. (d) and (f) shows the denoised chroma channels which have had their noise successfully removed. PSNR: (b) 37.68. ($\lambda = 0.5$)



(a) (b)

(c) (d)

(e) (f)

Figure 5: Impulse noise removal example. (a) shows an original colour image and (b) a noisy version that has had impulse noise added to the chroma channels in the YUV space. (c) is our reconstructed image which is virtually identical to the original. (d) is a typical result obtained using Neat Image or DenoiseMyImage. The impulse noise affecting the chroma is illustrated by (e) while the success of our algorithm for impulse removal is shown by (f). PSNR: (c) 42.20. ($\lambda = 0.5$)

Mimicking POV-Ray Photorealistic Rendering with Accelerated OpenGL Pipeline

J. Pečiva
Brno University of Technology
Božetěchova 2
612 66 Brno, Czech Republic
peciva@fit.vutbr.cz

P. Zemčík
Brno University of Technology
Božetěchova 2
612 66 Brno, Czech Republic
zemcik@fit.vutbr.cz

J. Navrátil
Brno University of Technology
Božetěchova 2
612 66 Brno, Czech Republic
inavrati@fit.vutbr.cz

ABSTRACT

Traditional ray tracing algorithms tend to provide photorealistic results but at high computing costs. Rendering times of minutes or days are not exceptional. On the other side, hardware accelerated OpenGL rendering can provide real-time interaction with virtual environment with unnoticeable rendering times. This paper attempts to bring these two together and attempts to give an answer on the difficulty of implementing real-time photorealistic rendering. The paper presents case study on mimicking of POV-Ray photorealistic rendering with accelerated OpenGL pipeline. The study shows the opportunity to accelerate some photorealistic algorithms by real-time approaches while, on the other side, it locates the parts that are difficult to replace by traditional real-time rendering paradigms. Particularly, it is shown how to implement primary and shadow rays and POV-Ray-like material model using accelerated OpenGL pipeline using modern shader technology. On the other side, the difficulties of reflected and refracted rays implementation using real-time rendering approaches is discussed.

Keywords
photorealistic rendering, POV-Ray, OpenGL, raytracing, shaders

1 Introduction

Photorealistic rendering is a very important domain in computer graphic because of the realism that is often expected from graphics applications. Beginnings of photorealistic rendering can be traced back to the '60s to the invention of the ray casting algorithm [Appel 1968], followed later by ray tracing algorithm [Whitted 1980]. Ray tracing and other methods of photorealistic rendering, however, tend to be computationally very expensive even for today's hardware, often forcing users to wait minutes, or even days for high quality results. On the other side, many applications of real-time computer graphics desiring for higher realism exist. They can not accept computationally expensive algorithms of photorealistic computer graphics. Their users require high productivity and interaction with the graphics scenes. Such requirements are difficult to achieve in non-real time applications.

Areas desiring real-time photorealistic rendering include, for instance, CAD and architecture applications for interior design. Architects often want to immediately see the esthetic value of the designed model. Presently, they are forced either to compromise productivity by waiting for the results of the photorealistic rendering, or to compromise visual quality by throwing away photorealism and by using standard approaches of real-time computer graphic instead. However, visual quality is often essential while standard approaches of real-time computer graphics usually tend to provide poor results on advanced scene lighting setups. If real-time photorealistic visualizations would be possible, they would provide high quality visualizations, making architects and designers much more time effective while increasing the final quality of the designed models. Other applications include visualizations for civil engineering, scientific research, computer games, and visualizations in general.

Recent enhancements in programability of the graphics processor, particularly shaders, gives a question whether it would be possible to implement and accelerate some photorealistic algorithms using standard approaches of real-time computer graphics. This paper does so by a case study of mimicking POV-Ray rendering with accelerated OpenGL pipeline while evaluating speed up factors of accelerated parts and describing difficulties with algorithms that are difficult to match with current OpenGL rendering paradigm.

2 State of the Art

The beginnings of photorealistic rendering go back to the '60s when ray casting was developed [Appel 1968]. Ray tracing [Whitted 1980] extends the idea of the ray casting by recursion – tracing the ray in the scene through several reflections. [Arvo 1989] presented a number of methods for ray tracing acceleration.

A number of modifications of ray tracing were developed. Path tracing [Kajiya 1986] uses Monte Carlo for stochastic evaluation of light distribution in the scene. It was extended to bidirectional path tracing [Lafortune 1993]. Photon mapping [Jensen 1995] [Jensen 2001] creates scene light distribution by shooting large number of photons from the light sources and making them bounce in the scene until they are absorbed somewhere, forming a photon map. To render a such scene, path tracing can be used, looking for contributions of closest photons at each surface hit.

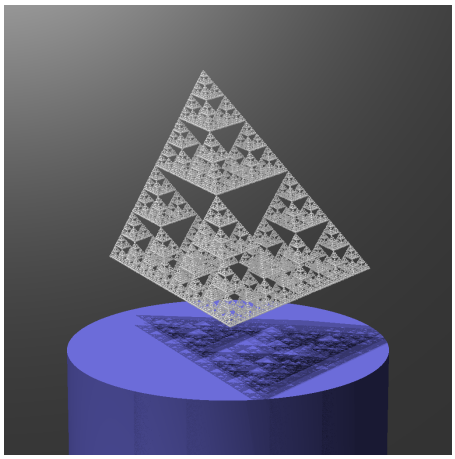


Figure 1: PRAY ray tracing tool rendering of fractally generated scene (4096 sphere objects in total) [Zemcik 1995]

Another path tracing optimization is Metropolis light transport [Veach 1997].

With the evolvement of ray tracing algorithms, various ray tracing software emerged, such as POV-Ray in 1989, Yafaray (2002), and Radiance ray-tracing software system [Larson 1998].

A number of real-time ray tracing attempts were made. REMRT/RT tools (based around BRL-CAD ray-tracer) are parallel network distributed ray-tracing system developed in 1986 and credited at Siggraph 2005 as a first real-time ray tracer capable of rendering several frames per second [Muuss 1987]. OpenRT (www.openrt.de) [Wald 2002] is a standalone closed-source real-time ray tracer. In 2006, Intel demonstrated the performance of its new dual-core processors on POV-Ray 3.7 beta that is capable of SMP (Symmetric Multiprocessing). In 2008, Intel demonstrated a special version of the computer game Enemy Territory: Quake Wars using ray tracing as rendering method. The demonstration used a 16-core system running at 2.93 GHz and providing 14-29 frames per second (FPS) on basic HD (720p) resolution [Valich 2008].

Some attempts were made for special ray tracing acceleration hardware. One of them was "Rendering on demand" [Chalmers 2006] using specialized hardware based on DSP and FPGA – (Field programmable gate array). The Saarland University developed another FPGA based system called Ray Processing Unit capable of accelerating ray tracing algorithms in hardware [Woop 2005]. The University of Utah developed a highly parallel multicore solution for real-time ray tracing [Spjut 2009].

Other approaches tried to use the ever-growing power of GPUs for ray tracing. [Purcell 2002] was the first one that presented ray tracing running on GPU. Photon mapping followed [Purcell 2003] with more complex ray tracing algorithms [Horn 2007][Gunther 2007][Shih 2009][Garanzha 2010]. Finally, Nvidia came with its own GPU-accelerated ray tracing API called OptiX [Nvidia 2009].

2.1 Real-time Rendering

Although ray tracing on GPU seems to be a promising approach, [Garanzha 2010] states that its GPU approach is only about 4x quicker than the CPU friendly implementation running on Core 2 Quad. Ray tracing algorithms as those mentioned here, are currently capable of processing about 100Mrays/second on scenes with 100K triangles (roughly said). It is still a level of magnitude behind the processing power of the traditional rendering pipeline as will be shown in the paper. Thus, number of attempts, such as [Loviscach 2004], were made to approach photorealistic rendering with the traditional OpenGL rendering pipeline.

Real time graphics used to focus mostly on performance and lacked the support for advanced lighting effects. Applications were forced to use precomputed lighting or a fixed lighting model [Wright 2004] that is too limiting for many modern graphics applications (see figures 2, 3). Invention of shaders [Rost 2005] for the OpenGL pipeline brought new possibilities for realism and lighting effects of real time rendering (see figure 4). Moreover, the shader programming capabilities grow with each new graphics card generation, giving more and more possibilities to map some photorealism algorithms to the real time rendering paradigms.

This paper is going to investigate the topic and find photorealism algorithms that can be accelerated and identify those that are difficult to map to the current OpenGL based real time rendering architectures. The photorealism will be studied on POV-Ray as it is a popular, recognized, and publicly available photorealistic ray tracer. Moreover, POV-Ray's source code is publicly available to be studied, modified, etc. Finally, the investigation will be done as a case study on mimicking POV-Ray with OpenGL using shader technology as this will easily show difficult algorithms to accelerate as well as performance gains of successfully implemented parts.

3 POV-Ray Architecture

POV-Ray will be divided into three main parts for the purpose of this paper:

- ray tracing rendering
- scene description and modelling library
- material modelling algorithms

Figure 2: Computer game Counter Strike (release year: 2000) Note static shadows of the buildings and shadows absence under moving people

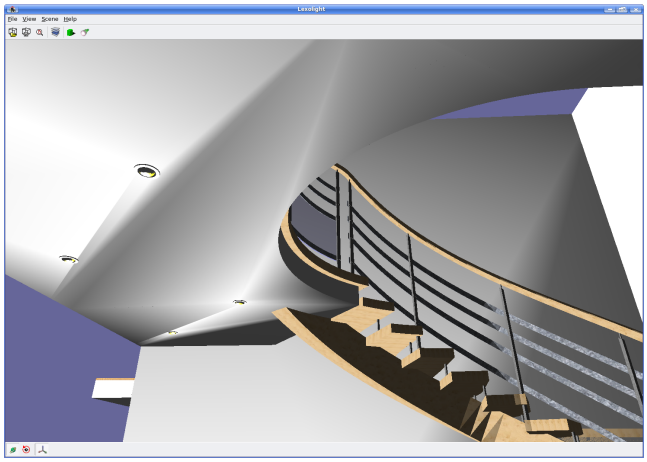


Figure 3: Lighting artifacts on a scene using standard OpenGL Gouraud shading

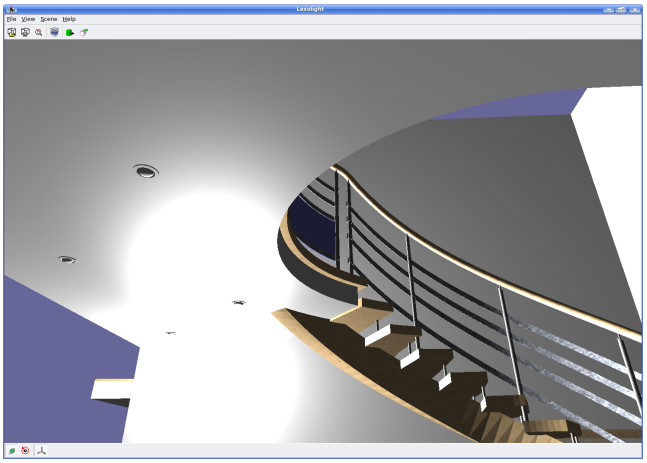


Figure 4: Per-pixel lighting in a scene rendered using shader technology

Only core functionality of these will be investigated, omitting for example, radiosity rendering and other various extensions.

POV-Ray ray tracing rendering is delivering photorealistic results on one side while requiring much computing resources on the other side. Rendering times of hours or days are not exceptional. Mimicking ray tracing in the OpenGL rendering pipeline is a challenging task as they are using an opposite approach: POV-Ray is tracing rays from the camera while OpenGL is projecting the scene geometry to the camera. This small difference brings various challenges to various light effects that will be discussed through the paper.

The scene description language of POV-Ray is very robust. It starts from triangle meshes and atmospheric effects and finishes with splines, blobs, run-time evaluated functions, and animations. In this paper, we will focus just on triangle meshes because that is the natural representation to OpenGL and all other representations can be tessellated – producing a triangle representation.

Modelling of materials is more advanced in POV-Ray than in standard OpenGL. It includes even multiple layers of material and procedurally generated material surfaces. Many of these advanced material approaches should be implementable in OpenGL shaders. This paper will not go to the excessive material functionalities of POV-Ray, rather mimicking of core material functionality will be discussed in section 3.2.

3.1 Ray Tracing

POV-Ray is a ray tracer. It casts a ray or a number of rays through each pixel of the image, rendered by camera. These rays are called primary rays. The ray travels through the scene until it hits a scene object, a light, or escapes the scene. If some object is hit, the object's material is processed and computed color is assigned to the ray. If it is primary ray, the ray's color is immediately assigned to the rays image pixel. However, depending on the material properties and the light setup, material processing may cause additional rays to be cast from the point of the object hit to various directions. Such rays are called secondary rays. When these rays hit other surfaces in the scene, these secondary rays may spawn recursively additional secondary rays. If the light is hit, it assigns the ray the light's color. If the ray leaves the scene, it is usually assigned the background's or sky's color.

POV-Ray assigns the level's number to the rays cast. The primary rays cast from the camera are level 0 rays. When a surface is hit by a ray of level n and it emits a new ray or several rays, they are said to be of level n+1. Tracking of the ray level avoids infinite recursions in some scenes and limits the rendering time by giving a maximum ray level limit. POV-Ray sets the limit to 5 by default while the user may increase it as needed, for example, on scenes with many mirrors.

3.2 Ambient Scene: Level 0 Rays

The idea of mimicking POV-Ray with the standard OpenGL pipeline leads immediately to the crucial question whether POV-Ray's ray casting can be replaced by the rendering pipeline. The approach of OpenGL is to project the scene's geometry to the camera's rendering plane and to change the pixel's colors as the geometry is rasterized and fragments are processed. POV-Ray's approach is the opposite. The rays are cast through the rendering plane and the color of the pixels are determined by the scene geometries hit by the rays. If only direct rays (e.g. level 0) are considered, both approaches should be interchangeable. The experiment of figures 5 and 6 proves the idea. Both images are of the same quality while POV-Ray's image took 4 seconds to render and the OpenGL's one 2.2 milliseconds (i7-920@2.66GHz, GeForce GTX 260). An acceleration factor of 1800 is the first achievement of this paper.

However, limiting rays to level 0 only prevents the scene geometry to be illuminated by light sources and only the ambient component of the light model affects visual appearance. To include light sources, rays of level 1 need to be introduced, providing the secondary rays cast from the intersection point to the light sources.

3.2 POV-Ray's Surface Model

To accelerate POV-Ray's level 1 rays in OpenGL, it is necessary to properly model POV-Ray's interaction of a ray with a surface. If the ray hits the object's surface any combination of four things might happen:

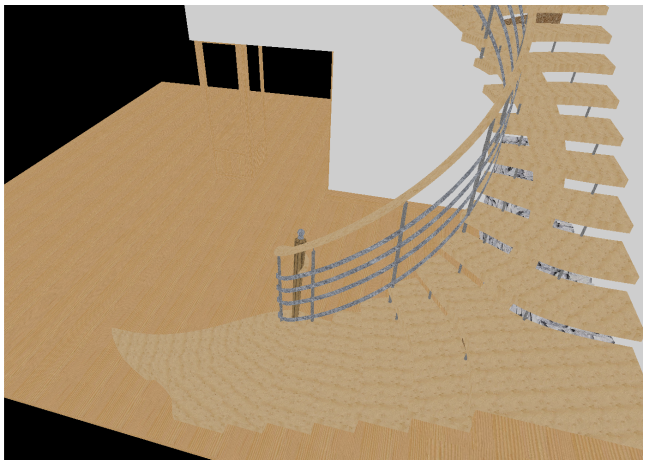


Figure 5: POV-Ray rendering using only level 0 rays

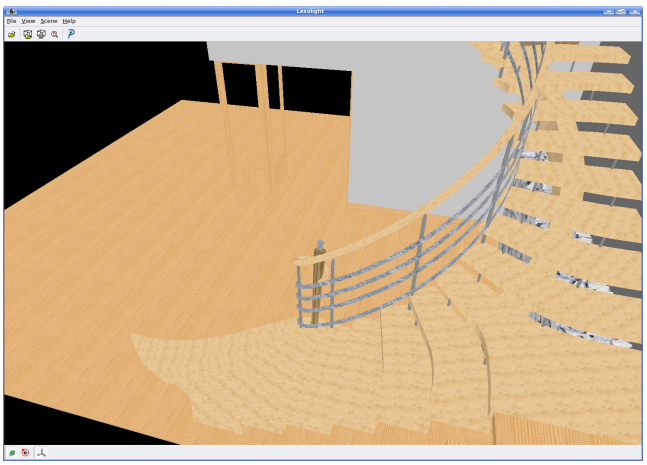


Figure 6: OpenGL rendering using ambient light

- absorption – the light ray or part of its intensity is absorbed, lowering its intensity and possibly altering its hue by absorption of some wavelengths only
- reflection – the light ray is reflected in one or more directions, secondary rays are cast
- refraction – transparent and translucent materials may cause a portion of the light ray to refract into the object
- fluorescence and emission – the surfaces may emit the light of different wavelengths, possibly altering the spectrum of the ray

POV-Ray models the surface absorption like OpenGL. It uses the RGB light color model. If the surface if purely blue, it absorbs all red and green component of incoming light rays. If it is 50% grey, half of the light ray's intensity is absorbed. Various reflection types are modelled by ambient, diffuse, specular, phong, and reflection material components. Refraction is modelled by the component of the same name, by the densities of refraction environments, and the amount of transparency of a given object. Fluorescence and emission effects are simulated by the ambient component or material components set in a way that the amount of outgoing light is bigger than the amount of incoming light. This may happen, for example, when one of the material components is bigger than 1.0.

However, modelling POV-Ray's material properties with OpenGL is not an easy task because the surface materials of POV-Ray are very robust and include material libraries and procedural functions. Because covering the material libraries and procedural functions would be time expensive work and would provide enough material for a paper dedicated just to this topic, this paper will be limited just to the core material modelling capabilities of POV-Ray.

POV-Ray calls the set of material surface properties "texture" (note the name collision with OpenGL's "texture"). The texture is composed of pigment, finish, and normal. The pigment can be a color, image map (e.g. 2D texture), or color map (procedural texture). Colors and image maps are the functionalities that have equivalents in OpenGL. Color maps are more complicated. They map floating point values in a range from 0.0 to 1.0 to color values specified in the map. The floating point values can be generated by various mapping functions and be modified by functions like turbulence and frequency. These procedural functions can be probably implemented using shaders, however they are out of scope of this paper.

Finish is the next item of POV-Ray's surface texture. Knowing the pigment, e.g. surface color at a given point – this may include texture mapping and filtering – finish gives the amount of ambient, diffuse, specular, phong, reflection, and refraction component. All these values are floats, usually between 0.0 and 1.0 to specify range from absence (value 0.0) to full intensity (1.0) of a given material component. Negative values and values over 1.0 can be used as well to create special or unrealistic effects. These intensities just multiply their value with the pigment color, resulting in the color of a particular component. Mimicking of all POV-Ray's finish material functionality in accelerated OpenGL is described summarized in the table 1.

The last POV-Ray's material surface property is normal. It is designed for surface normal manipulation. It can be specified by a bump map or a procedural approach. The procedural generators are out of the scope of this paper, so only bump maps are considered. The bump maps have mandatory support in OpenGL since version 1.3 [Segal 2001], thus they can be implemented.

3.3 Secondary Rays

After the investigation of primary rays (level 0) and color processing on POV-Ray's surface textures and seeing that it is possible to implement them in OpenGL, it is possible to come to the next step: POV-Ray's level 1 rays. When a level 0 ray cast from the camera hits a surface of an object, several secondary rays of level 1 may be cast. Secondary rays called shadow rays are sent to each light source to see whether anything is in the way and obscures the light source. If there is nothing in the way, the intersection point is illuminated by the particular light source and the color of the light source is assigned to the ray. If the light is obscured by an object, the black color is assigned to the ray instead.

If the surface has a non-zero reflection component, another secondary ray is cast in the direction of the reflection vector. If the surface is not completely opague, e.g. it is transparent or semi-transparent, another ray is cast into the object while a refraction effect may apply. The secondary rays are processed recursively in the same way as the primary rays, making secondary reflections and refractions, tertiary reflections, etc. At the end of each ray

POV-Ray's Finish component	Required rays to be cast	Finish component description	Implementability in OpenGL
Ambient	no ray casting required	pigment multiplied by ambient component and global ambient intensity	yes, using ambient color and global ambient light
Diffuse	secondary ray is cast to each light source to test its visibility	for each visible light source compute sum of diffuse equations (Lambertian reflectance [Phong 1973]) of the pigment multiplied by diffuse component, and color of the ray cast to the light source. Brilliance parameter may modify distribution of the reflection.	yes, but light source visibility requires accelerated shadow algorithms to be used. For example, shadow maps or shadow volumes techniques can be used. Per-pixel lighting requires color components to be computed using shaders because OpenGL's per vertex lighting provides often unacceptable results (see figure 3).
Phong		for each visible light source compute phong highlight [Phong 1973] of pigment, phong intensity, light source ray color. Pigment color affects the ray if metallic is specified.	
Specular		for each visible light source compute specular highlight [Blinn 1977][Phong 1973] of pigment, specular intensity and light source ray color. Pigment color affects the ray if metallic is specified.	
Reflection	one secondary ray cast in direction of reflection vector	color of reflection ray is multiplied with reflection intensity	OpenGL provides direct support for transparency, but robust implementation of reflection and refraction is not trivial. Possible solutions are discussed in the paper.
Transparency	one secondary ray cast into the object	the color of the ray is multiplied by transparent intensity. The direction of the ray may be affected by refraction	

Table 1: Relation between POV-Ray materials and OpenGL materials

level n evaluation, the color of all the rays of the level n+1 are taken and included in the color computation of the ray of the level n. Finally, when level 0 ray evaluation is completed, its color is assigned to the pixel of the image.

Visual results are shown in the figure 7 for POV-Ray while using level 0 and 1 rays only, and figure 8 for the OpenGL accelerated implementation.

3.4 POV-Ray Lights

Casting of secondary rays, particularly shadow rays cast towards light sources, is influenced by the light type. As can be expected, POV-Ray uses a more advanced lighting model than built-in OpenGL lights. Anyway, the limited built-in light model was made obsolete in OpenGL 3.0 and programmers are encouraged to create their own lighting effects using shaders.

POV-Ray uses several types of lights: point light, parallel light (like OpenGL's directional light), spot light, cylindrical light, and area light. The first three have equivalents in OpenGL, the fourth should be possible to implement in shaders and the last one – area light – can be implemented as an array of point lights. That is actually the way that area light is implemented in POV-Ray. All the details of POV-Ray's lights and their implementability by OpenGL are described in the table 2. Implementability was verified for point light, directional light and spot light. Cylindrical light is rarely used, and area light is just the array of point lights in POV-Ray. To prove implementability of the lights, point light, directional light and spot light were implemented in OpenGL's shaders. The results are shown in the figures. Because of the space limitations, we show just the spot light scene as the spot light is the most complex type when considering mentioned light types and their POV-Ray implementation. A summary of lights implementability is shown in the table 2.

3.5 Higher Level Rays

When we attempted to analyze implementability of level 2 rays in OpenGL, we found it very difficult, particularly when considering the general case. Following the goals set in the beginning of this paper, this section successfully identifies one particular difficulty: to mimics POV-Ray rays starting from level 2. Additionally, the section is going to discuss available options that should be addressed in the future.

Rays of higher level starting from level 2 bring very cool effects, like mirroring, reflections and refractions. One POV-Ray example is shown in the figure 9. However, these rays are not straightforward to implement for the general case. Some GPU approaches follow that can be used to reach similar effects as those created by level 2 rays:

- scene mirroring – there are various approaches to implement mirrors in OpenGL. Usually, they just duplicate the scene behind the mirroring surface while using stencil test to limit rendering just to mirroring surface.

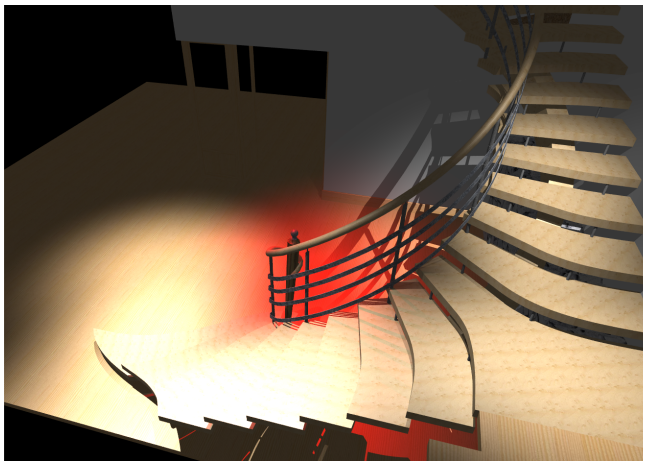


Figure 7: POV-Ray rendered image using rays level 0 and 1

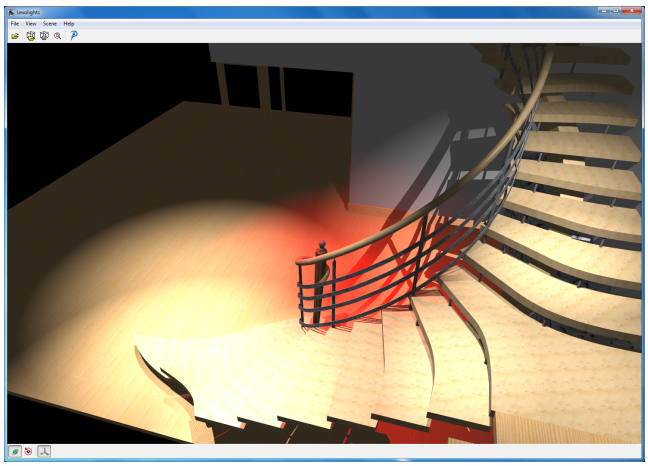


Figure 8: OpenGL rendered image using per-pixel lighting

- render to texture – the mirrored or reflected geometry can be pre-rendered to the texture that is applied to the surface afterwards.

- environment mapping – similar to render to texture approach, but it usually renders all surrounding environment (360 degrees) to, for example, cube map. The pre-rendered environment is then mapped to the geometry using, for instance, texgen OpenGL functionality or shaders. Effects like mirrors or surface reflections are easily implementable using environement mapping.

- GPU raytracing – there were number of attempts to perform ray tracing on GPU, such as [Garanzha 2010][Shih 2009] [Horn 2007][Gunther 2007].

The first three OpenGL-based options seem appropriate to create nice reflections and refraction effects on planar surfaces. However, they are breaking projection coherency that was present with primary rays and shadow level 1 rays. Breaking of this coherency may result in a huge number of projections and pre-rendering to texture. Theoretically, each surface that is not coplanar with any other surfaces may require its own projection. Such solution may turn to be very performance expensive. Moreover, curved surfaces, such as NURBS, exhibit even more problems. They can not be processed directly as they are not planar surfaces. They can be tesselated, but to reach high quality visual results, too many not coplanar surfaces may be produced, possibly harming performance too much.

Another approach can be to start GPU ray tracing at level 2 rays. Although GPU ray tracing still does not reach the performance of

POV-Ray's light type	Description	Implementability in OpenGL
Point light	The light placed in the scene shining equally in all directions	Similar to OpenGL's point light except that OpenGL's light does not cast shadows by default. One of shadow techniques need to be utilized. Lighting should be implemented per-pixel, for example, in shaders.
Parallel light	The light whose rays come in parallel in certain direction	Similar to OpenGL's directional light with the exception of shadows. One of shadow techniques need to be used.
Spot light	Like the point light but the light is restricted to a cone in some direction. The light intensity in the cone can be modulated. POV-Ray uses radius, falloff, and tightness parameters.	Similar to OpenGL's spot light except shadows and light intensity modulation. Shadows need to be implemented. Intensity modulation inside the cone is different from the OpenGL's built-in spotlight. However, shaders are usually used to implement per-pixel lighting and POV-Ray's light modulation can be computed there as well.
Cylindrical light	Like spot light but it is constrained by a cylinder. It is useful for effects light laser beams. Technically, it is not based on parallel light as could be expected, but on point light whose rays are constrained by the cylinder.	Can be implemented in OpenGL shaders. Shadow technique needs to be used.
Area light	Finite 1D or 2D rectangular area providing flat panel light. Technically, it is implemented using array of point lights. As a side effect, the light provides soft shadows.	Can be implemented as an array of point lights. Alternatively, some area light model [Au 2007] implemented using shaders can be used for instance.

Table 2: POV-Ray's lights and their implementability by OpenGL

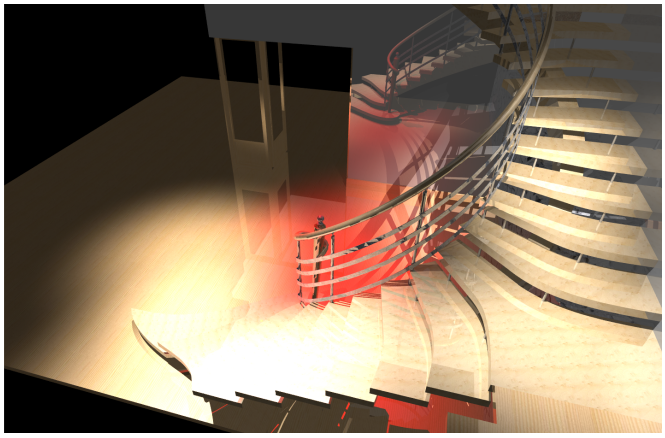


Figure 9: POV-Ray rendered image using rays level 0,1, and 2

the standard OpenGL rendering paradigm, it may outperform OpenGL approaches for level 2 rays. Further investigation would be necessary to clarify the best approach that may even lead to a hybrid solutions – rendering big planar surfaces in OpenGL and ray tracing the small or curved surfaces.

4 Experiments

All the algorithms developed in this paper were tested as a part of our Lexolights open source project available at: http://lexolight.sourceforge.net. The website includes win32 binaries covering the functionalities mentioned in this paper. Namely, all POV-Ray core material functionality is implemented, e.g. all major elements of POV-Ray's texture finish, with exception of reflection and refraction elements that would require support of level 2 rays. Next covered area is lights that include point, directional, and spot light while we used shadow maps [Williams 1978] and LiSPSM [Wimmer 2004] for shadow rays visibility tests. Lexolight is implemented using OpenSceneGraph (http://www.openscenegraph.org) – high level rendering library built on the top of OpenGL. As some of our algorithms were better suited to be included directly in OpenSceneGraph, such as POV-Ray scene converter and exporter, we submitted them to be included in the upcoming release for the profit of the open source community.

4.1 Performance

To evaluate the performance gains, several measurements were made for POV-Ray rendered scenes. Then, the same scene was rendered by hardware accelerated OpenGL. The results are summarized in the table 3. We used the scene composed of approximately 83000 triangles visualized on high and low performance CPUs and a variety of graphics cards ranging from hi-end, through mobility versions, to old low-end GPUs. POV-Ray rendering took from few seconds to about a minute. Rendering of the same scene for level 0 and level 1 rays using POV-Ray's accelerated OpenGL approach took less than 150ms even on very old mobile fill-rate limited graphics card. For nowadays hi-end graphics cards, the speed up factor stayed far above 1000. We consider such speed up an interesting result. It shows a potential to accelerate level 0 and level 1 rays of ray tracers. Another option can be to further investigate acceleration of level 2 rays or to consider whether a hybrid solution would be the best option for close-to-photorealistic real-time rendering.

5 Conclusions

This paper investigated the realization of photorealistic rendering in real-time, accelerated by the graphics hardware. The investigation was made on a case study by mimicking of POV-Ray with OpenGL's rendering paradigm accompanied with the latest programmable shader technologies.

The presented study proved that the core POV-Ray material and lighting functiontionality can be implemented in accelerated OpenGL. It turned out that it is very easy to accelerate primary rays (e.g. level 0 rays) cast by POV-Ray when rendering the scene and a speed up factor of 400 was measured even on very old hardware. Secondary rays of level 1 were accelerated as well while together with level 0 rays, the speed up factor was over 3000 for modern hardware. Rays of level 2 and higher turned out to be difficult to be implemented in accelerated OpenGL. They may require additional research efforts and using of advanced rendering techniques as was discussed in section 3.5.

Future research should extend acceleration of rays of level 0 and 1 to higher level rays. Although it may be difficult to do so, it would enable additional visual effects and may even lead to ideas of accelerating global illuminations methods, such as radiosity.

	Level 0 Rays Acceleration			Level 1 Rays Acceleration			Level 2 Rays
	POV-Ray rendering (level 0 rays)	OpenGL ambient rendering	Speed-up factor	POV-Ray rendering (level 1 rays)	OpenGL per-pixel ligting	Speed-up factor	POV-Ray rendering (level 2 rays)
i7-920 @2.66GHz, GeForce GTX 260	4s	2.2ms	1800	14s	4.4ms	3200	21s
Core 2 Duo @ 2.00GHz, Radeon HD 3670 Mobility	8s	7.4ms	1100	21s	18ms	1200	33s
Athlon XP 2000+, Radeon HD 2600 XT	12s	11.4ms	1100	37s	20.5ms	1800	58s
Core 1 Duo @ 1.83GHz, Radeon X1300M	13s	33ms	400	36s	133ms	270	55s

Table 3: Performance comparison
(screen size: 1440x1050, one omnidirectional light, 83000 triangles)

Acknowledgements

This work was supported by the Ministry of Education, Youth and Sports of the Czech Republic under the research program LC-06008 (Center for Computer Graphics). Special thanks to Cadwork Informatik AG and Cadwork development team in Brno for the support of this project.

References

AU, A. 2007. A simple area light model for GPUs. In *Shader X5*, W. Engel, Ed. Charles River Media, Chapter 2.1, 63—67.

APPEL, A. 1968. Some techniques for shading machine renderings of solids. In *Proceedings of the April 30--May 2, 1968, Spring Joint Computer Conference* (Atlantic City, New Jersey, April 30 - May 02, 1968). AFIPS '68 (Spring). ACM, New York, NY, 37-45. DOI= http://doi.acm.org/10.1145/1468075.1468082

ARVO, J. AND KIRK, D. 1989. A survey of ray tracing acceleration techniques. In An introduction To Ray Tracing, A. S. Glassner, Ed. Academic Press Ltd., London, UK, 201-262.

BLINN, J. F. 1977. Models of light reflection for computer synthesized pictures. SIGGRAPH Comput. Graph. 11, 2 (Aug. 1977), 192-198. DOI= http://doi.acm.org/10.1145/965141.563893

CHALMERS, A., DEBATTISTA, K., GILLIBRAND, R., LONGHURST, P., AND SUNDSTEDT, V. 2006. Rendering on demand. In EGPGV2006 - 6th Eurographics Symposium on Parallel Graphics Visualization, Eurographics, 9--18.

GARANZHA, K., LOOP, C., 2010. Fast Ray Sorting and Breadth-First Packet Traversal for GPU Ray Tracing, Computer Graphics Forum 29, 2. Proceedings of Eurographics 2010, Norrköping, Sweden.

GUNTHER, J., POPOV, S., SEIDEL, H., AND SLUSALLEK, P. 2007. Realtime Ray Tracing on GPU with BVH-based Packet Traversal. In Proceedings of the 2007 IEEE Symposium on interactive Ray Tracing (September 10 - 12, 2007). IEEE/Eurographics Symposium on Interactive Ray Tracing. IEEE Computer Society, Washington, DC, 113-118. DOI= http://dx.doi.org/10.1109/RT.2007.4342598

HORN, D. R., SUGERMAN, J., HOUSTON, M., AND HANRAHAN, P. 2007. Interactive k-d tree GPU raytracing. In *Proceedings of the 2007 Symposium on interactive 3D Graphics and Games* (Seattle, Washington, April 30 - May 02, 2007). I3D '07. ACM, New York, NY, 167-174. DOI= http://doi.acm.org/10.1145/1230100.1230129

JENSEN, H. W., CHRISTENSEN, N. J. 1995. Photon maps in Bidirectional Monte Carlo Ray Tracing of Complex Objects". *Computers & Graphics* 19 (2), pages 215—224.

JENSEN, H. W. 2001. Realistic Image Synthesis Using Photon Mapping. A. K. Peters, Ltd.

KAJIYA, J. T. 1986. The rendering equation. SIGGRAPH Comput. Graph. 20, 4 (Aug. 1986), 143-150. DOI= http://doi.acm.org/10.1145/15886.15902

LAFORTUNE, E.P. AND WILLEMS, Y.D., Bi-directional Path Tracing, Computer Graphics Proc., Alvor (Portugal), 1993, pp. 145-153.

LARSON, G. W. AND SHAKESPEARE, R. 1998 *Rendering with Radiance: the Art and Science of Lighting Visualization*. Morgan Kaufmann Publishers Inc.

LOVISCACH, J. 2004 Emulating an Offline Renderer by 3D Graphics Hardware, WSCG 2004, 269-276.

MUUSS, M. J. 1987. RT & REMRT: Shared Memory Parallel and Network Distributed Ray-tracing Programs. *Proceedings of 4th Computer Graphics Workshop*, Cambridge, MA, USA, October 1987. Usenix Association, pp 86-98.

NVIDIA 2009. *Nvidia OptiX*. Nvidia website: http://www.nvidia.com/object/optix.html.

PHONG, B. T. 1973 *Illumination for Computer-Generated Images.* Ph.D. Thesis. UMI Order Number: AAI7402100., The University of Utah.

PURCELL, T. J., BUCK, I., MARK, W. R., AND HANRAHAN, P. 2002. Ray tracing on programmable graphics hardware. In Proceedings of the 29th Annual Conference on Computer Graphics and interactive Techniques (San Antonio, Texas, July 23 - 26, 2002). SIGGRAPH '02. ACM, New York, NY, 703-712. DOI= http://doi.acm.org/10.1145/566570.566640

PURCELL, T. J., DONNER, C., CAMMARANO, M., JENSEN, H. W., AND HANRAHAN, P. 2003. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (San Diego,California, 2003). Eurographics Association, Aire-la-Ville, Switzerland, 41-50.

ROST, R. J. 2005 *Opengl(R) Shading Language (2nd Edition)*. Addison-Wesley Professional.

SEGAL, M., AKELEY, K. 2001. *The OpenGL Graphics System: A Specification (Version 1.3)*. Silicon Graphics, Inc. Available at: http://www.opengl.org/documentation/specs/

SHIH, M., CHIU, Y., CHEN, Y., AND CHANG, C. 2009. Real-Time Ray Tracing with CUDA. In Proceedings of the 9th international Conference on Algorithms and Architectures For Parallel Processing (Taipei, Taiwan, June 08 - 11, 2009). A. Hua and S. Chang, Eds. Lecture Notes In Computer Science, vol. 5574. Springer-Verlag, Berlin, Heidelberg, 327-337. DOI= http://dx.doi.org/10.1007/978-3-642-03095-6_32

SPJUT, J., KENSLER, A., KOPTA, D., AND BRUNVAND, E. 2009. TRaX: a multicore hardware architecture for real-time ray tracing. Trans. Comp.-Aided Des. Integ. Cir. Sys. 28, 12 (Dec. 2009), 1802-1815. DOI= http://dx.doi.org/10.1109/TCAD.2009.2028981

VALICH, T. 2008. *Intel converts ET: Quake Wars to ray tracing*. TG Daily. (June 12, 2008), Available online at http://www.tgdaily.com/html_tmp/content-view-37925-113.html .

VEACH, E. AND GUIBAS, L. J. 1997. Metropolis light transport. In Proceedings of the 24th Annual Conference on Computer Graphics and interactive Techniques International Conference on Computer Graphics and Interactive Techniques. ACM Press/Addison-Wesley Publishing Co., New York, NY, 65-76. DOI= http://doi.acm.org/10.1145/258734.258775

WALD, I., BENTHIN, C. AND SLUSALLEK, P. 2002. *OpenRT – A Flexible and Scalable Rendering Engine for Interactive 3D Graphics.* Technical report, Saarland University. Available at http://graphics.cs.uni-sb.de/Publications.

WHITTED, T. 1980. An improved illumination model for shaded display. *Commun. ACM* 23, 6 (Jun. 1980), 343-349. DOI= http://doi.acm.org/10.1145/358876.358882

WILLIAMS, L. 1978. Casting curved shadows on curved surfaces. SIGGRAPH Comput. Graph. 12, 3 (Aug. 1978), 270-274. DOI= http://doi.acm.org/10.1145/965139.807402

WIMMER, M., SCHERZER, D., PURGATHOFER, W. 2004. Light Space Perspective Shadow Maps, In Rendering Techniques 2004 (Proceedings Eurographics Symposium on Rendering), p. 143-151. June 2004.

WOOP, S., SCHMITTLER, J., AND SLUSALLEK, P. 2005. RPU: a programmable ray processing unit for realtime ray tracing. In ACM SIGGRAPH 2005 Papers (Los Angeles, California, July 31 - August 04, 2005). M. Gross, Ed. SIGGRAPH '05. ACM, New York, NY, 434-444. DOI= http://doi.acm.org/10.1145/1186822.1073211

WRIGHT, R. S. AND LIPCHAK, B. 2004 *OpenGL Superbible (3rd Edition)*. Sams.

ZEMCIK, P., CHALMERS, A. G. 1995. Optimised CSG Tree Evaluation for Space Subdivision. Computer Graphics Forum, p. 139-146, Netherlands, 1995

Automated 3D Visualization of Electron Microscope Tomograms

Chikara Yamashita
Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
573-0196, Japan
yamashita@is.oit.ac.jp

Koji Nishio
Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
573-0196, Japan
nishio@is.oit.ac.jp

Ken-ichi Kobori
Osaka Institute of Technology
1-79-1, Kitayama
Hirakata, Osaka
573-0196, Japan
kobori@is.oit.ac.jp

ABSTRACT

In this paper, we propose a 3D visualization method for ultra-high voltage electron microscope tomography intended for use with biological samples. The most important process for constructing 3D images from UHVTEM tomograms is the extraction of contours from 2D sliced images. However, automatic extraction of contours is difficult because of typical noise and artifacts. The proposed method automatically extracts contours by the 3D level set method. In general, the result of the extraction with the level set method depends on the definition of initial contours and parameters. These parameters are usually set manually. Our method automatically generates the initial contours and decides the fittest parameters for the level set method. We verify the effectiveness of our method by applying the technique to two types of unicellular organisms, to compare the results of the proposed method with manual extraction. The automated method successfully identified most cell tissues, with the exception of a limitation in the imaging of tubular-shaped cell structures.

Keywords

Electron microscope tomography, level set method, visualization, segmentation, contour extraction.

1. INTRODUCTION

Ultra-high voltage transmission electron microscopy (UHVTEM) has become a major contribution to medial imaging, because it permits the examination of intracellular structures, which are difficult to examine by conventional X-rays, computed tomography (CT), and magnetic resonance imaging (MRI) techniques [Joa06].

Three-dimensional visualization method can be classified into volume rendering and surface rendering. We use surface rendering to observe object surface. It is possible to extract the three-dimensional structure of cellular structures from UHVTEM data; however, 3D contours are difficult to generate automatically because of noise and artifacts in the data. The traditional approach for generating 3D contours of cellular structures traces the structures on successive two-dimensional slices of a three-dimensional reconstruction. Many hours are required, when the number of sliced images increases.

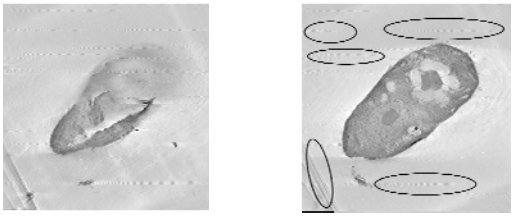
Our proposed method automatically extracts 3D contours using the level set method [Set99] [Sta03]. Generally, the extraction with the level set method requires the definition of the initial contours and the fittest parameters, which are difficult to define manually. The proposed method automatically generates the initial contours and decides the fittest parameters, thus expediting the process of 3D contour generation. We expect that this refinement of the level set method will facilitate the analysis and recognition of cell pathologies, with applications in a broad spectrum of biomedical disciplines.

2. FEATURES OF ULTRA-HIGH VOLTAGE ELECTRON MICROSCOPE TOMOGRAMS

Ultra-high voltage Electron microscope tomograms are generated by UHVTEM images, obtained on samples with thicknesses in the range of 0.1–10 μm. These thicknesses are sufficient to encompass the size of cell organelles and to permit the quantitative analysis of variations in shape and organization of pathological structures. One of the requirements of

the UHVTEM image is that the missing domain is minimized to enhance the precision of tomographic processing.

A transmitted image is obtained by inclining the sample irradiate the transmitted beam. A missing domain occurs in the resulting image because of mechanical limitations related to rotation of the sample. For example, the reconstructed image in Figure 1(a) reveals decreased resolution in the direction of the missing domain, indicated by the blurred, dim region. Also, the gold particles used in Ultra-High voltage electron microscopy for image alignment and registration appear as granular lines in UHVTEM tomograms; see Figure 1(b).



(a) **(b)**

Figure 1. Typical noise in ultra-high voltage electron microscope tomograms: (a) Blurred region reflects low S:N ratios; **(b) Ellipses identify shadows of gold particles.**

3. LEVEL SET METHOD

The level set method is commonly used in medical imaging because it successfully extracts blurred contours [Tak03] [Yos05]. The method is considered an active contour model because it allows topological change. The algorithm extracts contours based on a boundary condition where a region becomes a zero value in high-dimension function. The user defines an initial contour and the level set function which is decided from the initial contour. The algorithm then updates the level set function by solving partial differential equation. The image processing is complete when the variation in the level set function falls below a certain threshold value.

One problem with the level set method is that errors are accumulated at each update of the level set function, causing instabilities in the solution. To achieve a stable solution, the function must be "re-initialized", however, the processing cost of re-initialization is high [Cat03] [Chu05] and various faster methods have been introduced. In our proposed model we use the method of references [Chu05] which is a level set method without re-initialization. In the method of references, the update level set function φ is

$$\phi_{t+1} = \phi_t + rF(\phi) \quad (1)$$

where

$$F(\phi) = \mu \mathrm{P}(\phi) + \lambda L_g(\phi) + \nu A_g(\phi) \quad (2)$$

F is the speed function (the sum of the internal and external energies), r is the weight of the speed function, P is the internal energy function (the effect of re-initialization at the time of the update of the level set function is included), L_g gives the length of the zero isosurface of φ, A_g is introduced to accelerate the update of the level set function,

μ is the weight of the internal energy function, and λ and ν are the weights of the external energy function; if ν is positive, a contour moves in the shrinking direction; otherwise, a contour moves in the expanding direction.

In the level set method, values for the parameters of the level set function are set by experience, usually requiring a number of trials to establish values that give a suitable solution. The model proposed here eliminates these trials by establishing the parameters automatically.

4. 3D VISUALIZATION METHOD

In general, the technique of 3D visualization relies upon laminating the contours of 2D sliced images. When extracting a contour manually, the top and bottom images are established as references. In our proposed automated protocol, the top and bottom images are inputs to the three-dimensional level set method. The level set method requires the input of an initial contour; however, this contour is difficult to set manually. The initial parameters of the level set parameter are also difficult to set, because of the number and variety of cellular structures often embedded in a tomographic image. The proposed automated model sets the initial contour and level set parameters according to procedures in Figure 2.

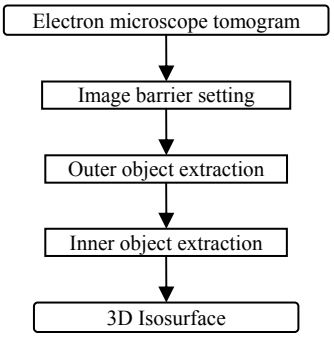


Figure 2. Overview of the proposed automated method.

Step 1) To reduce the influence of the missing domain that is characteristic of ultra-high voltage electron microscope images, an image barrier is set for volume data. The image barrier is discussed in Section 4.3.

Step 2) Extract the contour of outer object using the three-dimensional level set method.

Step 3) Extract the contour of inter object using the three-dimensional level set method.

Step 4) Step 3 is repeated as many times as necessary for each of the varieties of cellular tissue.

Step 5) Isosurfaces are generated using Marching Cubes [Wil87] for the results provided by Steps 2 and 3.

In addition, the proposed method automatically sets an initial contour and parameters for the level set method based on a reference region, as discussed in Section 4.2.

4.1 Definition of tissue boundaries

In order to reduce the processing time in samples with a variety of cell tissue types, the proposed method extracts the contour border of the outer tissue structure, and next extracts the contours of inner tissue structures. This hierarchical extraction technique reduces the processing required for calculations. We define structure of the cell group border of the biotissue as "inner tissue", and other contour and inside domain as "outer tissue". Figure 3 shows the relationship between inner tissue and outer tissue.

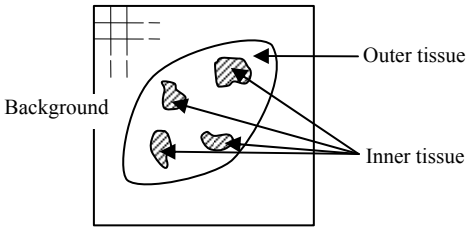


Figure 3. Tissue extraction.

Figure 4 shows the automated procedure to set an initial contour at the border of the outer domain structure. The contour, initially set at the boundary of the image, shrinks to conform to the outer tissue boundary. To set an initial contour on an inner domain structure, the initial contour is defined within the structure, and the contour expands to conform to the inner boundary of the domain.

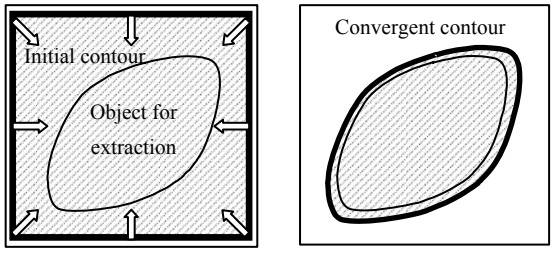


Figure 4. External tissue extraction: (a) Initial contour; (b) Result of extraction

4.2 Automatic initial contour generation

The extraction of the outer boundary of the "outer tissue" is relatively straightforward because the background pixel values are approximately constant. The initial contour for outer tissue can therefore be established either manually or automatically. However, it is difficult to establish an initial contour for inner tissue structures because of the number and variety of associated shapes, textures, brightness intensities, colors, etc. When the inner contours are extracted manually, the operator considers tissues with similar colors as the basis for organization, even if the shape of these structures varies.

Therefore, the proposed method automatically generates the initial contours for inner tissue structures based on color information within the sample. Figure 5 gives an overview of the automated process for initial contour generation.

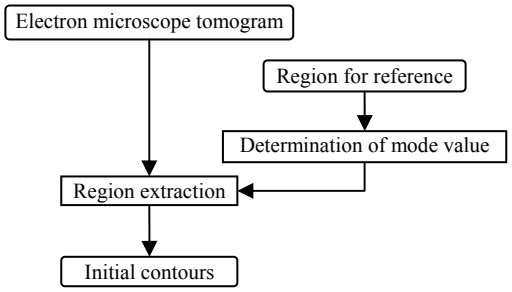


Figure 5. Overview of the automated process of initial contour generation.

Step 1) Choose an image nearly the center of volume data and select a region for contour extraction. Selected region defines the "region for reference"; see Figure 6(a).

Step 2) Determine the mode of brightness values in the region for reference.

Step 3) Determine the extraction range of brightness values for inner tissue

Step 4) To remove noise and artifacts, perform shrinkage processing on the Step 3 result.

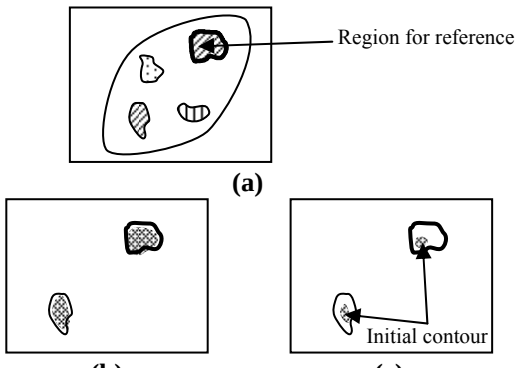


Figure 6. Automatic initial contour generation: (a) Region for reference; (b) Result of region extraction; (c) Initial contour.

If the frequency of brightness values within the inner tissue structure is approximately uniform, then the peak of the distribution defines the mode. The

process chooses a range of brightness values for extraction, and the extraction range encompasses the mode value; see Figure 7.

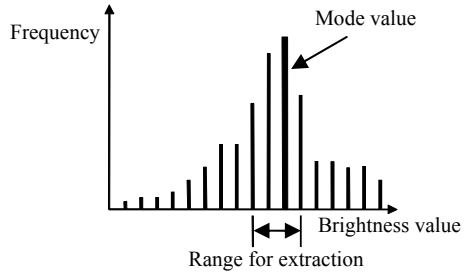


Figure 7. Extraction range.

The extraction range establishes the contrast used by the procedure to define the initial contour. Depending on the extraction range, the procedure may extract pixels that are outward of the actual boundary. In this case, one applies shrinkage processing on the extracted domain in Step 2.

In the proposed method, Steps 2–4 are performed for the cell structures within the outer tissue; repetition of this processing technique extracts the individual elements and establishes the organization framework of the tissue.

4.3 Setting of image barriers

For regions in which a low gradient intensity is included in the image, contours are extracted by the level set method. A contour is defined which extends inside of the cellular structure; see Figure 8(a).

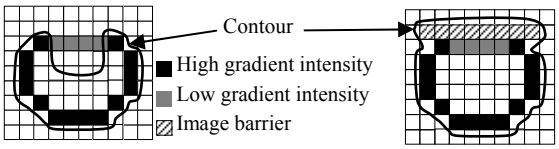


Figure 8. Image barrier (a) before setting and (b) after setting.

In the case of ultra-high voltage electron microscope tomography, missing domains create the region of the low gradient intensities. Missing domains are typically common on upper and lower boundaries. Thus, the application of the three-dimensional level set method may result in contours which extend into the structure. In addition, in the case of a tubular structure oriented in the section direction, a contour may extend into the inner regions of the tissue because the top and bottom boundary surfaces are not distinct. Therefore, the proposed method sets a barrier for the contour by defining the boundary of the structure (Figure 8(b)). This barrier is defined as an "image barrier."

The image barrier is set at the top and bottom ends of the domain manually; see Figure 9. The rectangular domain which encloses the cell tissue is included in these images. These images can them be replaced with the original input image.

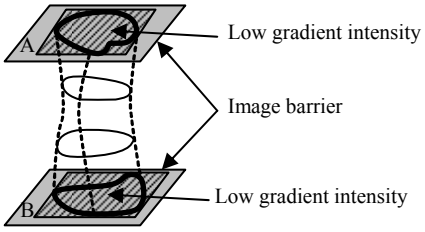


Figure 9. Image barriers.

4.3 Automatic parameter setting

The proposed method sets the parameters for level set method automatically for inner tissue levels. To decrease the processing time, the automated method processes a small domain and translates that domain to the whole image.

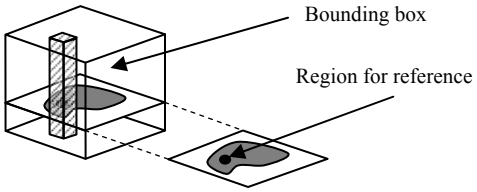


Figure 10. Bounding box generation.

A 3D bounding box is established outside the parcel of inner tissue which includes the region for reference; see Figure 10. The bounding box is elongated in a direction perpendicular to the plane of the region for reference. The initial contour is set using the level set method in this bounding box. The parameters are established when the contour corresponds to the boundary of the region for reference; see Figure 11(a). If the initial contour is inside of the region for reference, the parameters λ and v are increased and the level set method is reapplied; see Figure 11(b). If the initial contour is outside of the region for reference, the parameters λ and v are decreased and the level set method is reapplied; see Figure 11(c).

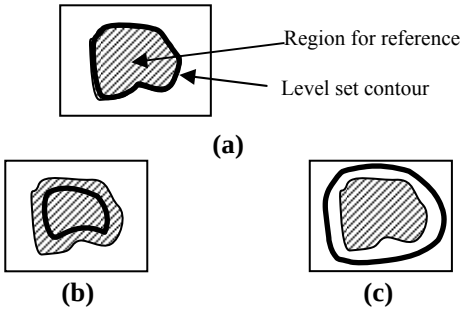
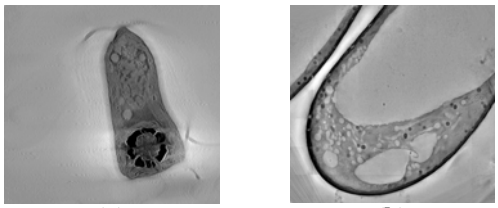


Figure 11. Evaluation of parameters: Fitting the level set contour to the region for reference (a) at the boundary, (b) inside the boundary, and (c) outside the boundary.

5. RESULTS

In our results, we compare the proposed automated method with the manual method of operation. We performed the automatic generation of the initial contour and three-dimensional level set method parameters based on the input of an initial contour.

Figure.12 shows ultra-high voltage electron microscope tomograms used to test the automated method: (a) *Dunaliella parva* (870 × 791 × 146); (b) *Brassica rapa* (512 × 496 × 101).



(a) **(b)**

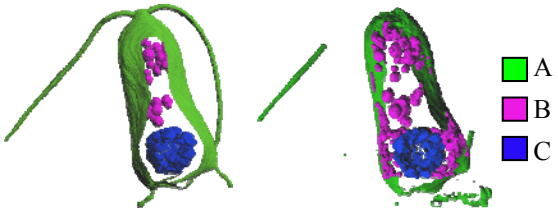
Figure 12. Ultra-high voltage electron microscope tomograms of: (a) *Dunaliella parva*; (b) *Brassica rapa.*

Figures 13 and 14 show the results of 3D visualizations obtained by (a) manual extraction and (b) the proposed automated method. The regions in the visualizations are referred to as outer tissue (regions A and D), granular inner tissue (regions B and E), and "other" inner tissue (regions C and F).

We assessed the effectiveness of the proposed method by evaluating whether contours established by manual extraction are also established by the proposed method. The ratio P of false detections to manually established contours is

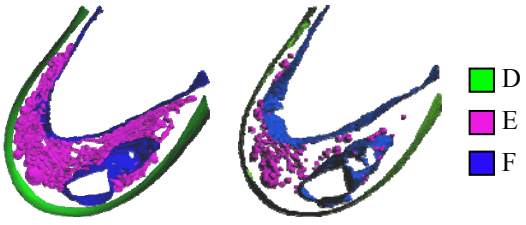
$$P = \frac{Negative\_error + Possitive\_error}{V\_manual}. \qquad (3)$$

V_manual is the number of voxels in the region of manual extraction. *Negative_error* is the number of voxels in regions of cellular tissue that the proposed method failed to extract. *Positive_error* is the number of voxels that the proposed method erroneously extracted from regions that were not part of the cellular tissue. P is 0 when the results of the automated and manual extractions are equal.



(a) **(b)**

Figure 13. Results for *Dunaliella parva*: (a) Manual extraction; (b) Proposed method.



(a) **(b)**

Figure 14. Results for *Brassica rapa*: (a) Manual extraction; (b) Proposed method.

Tables 1 and 2 show the evaluation values and processing time and the number of *V_manual*, *Negative_error* and *Positive_error* voxels, along with their relative contributions (in percentages) to the value of the evaluation parameter P for the 3 regions, in *Dunaliella* and *Brassica*, respectively.

	A	B	C
Evaluation value	0.133	1.464	0.242
Time (hour)	7.0	3.0	1.5
V_manual	13,349,695	449,990	942,446
Negative_error	616,980 4.6%	73,128 16.2%	105,993 11.2%
Positive_error	1,159,883 8.7%	585,851 130.2%	122,052 13.0%

Table 1. Evaluation values and processing time, and voxel values obtained for manually extracted contours, and corresponding errors associated with the automated extraction method; data for *Dunaliella.*

	D	E	F
Evaluation value	0.028	0.910	0.036
Time (hour)	13.2	3.0	15.8
V_manual	14,538,179	890,216	8,466,293
Negative_error	1,240 0.0%	547,500 61.5%	34,231 0.4%
Positive_error	402,573 2.7%	262,993 29.5%	273,441 3.2%

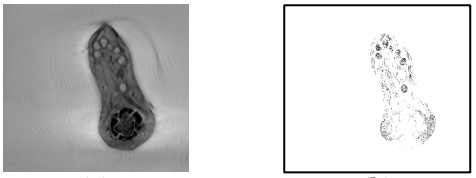
Table 2. Evaluation values and processing time, and Voxel values obtained for manually extracted contours, and corresponding errors associated with the automated extraction method; data for *Brassica.*

Evaluation values obtained for outer tissues (A and D) are less than 0.15 (Tables 1 and 2). Because outer tissues include a large proportion of high gradient intensity regions, which facilitate tomographic processing, it appears that contours can be readily extracted even if the initial contour is set roughly.

The evaluation values for "other" inner tissue regions (C and F) are 0.242 and 0.036 for the 2 trials (Tables 1 and 2). It is thought that initial contour generates cellular tissue in neighborhood. Because ratio these tissue in input tomogram is big.

The evaluation values of granular inner tissues (B and E) are relatively large (1.464 and 0.910; Tables 1 and 2). The *Positive_error* of B is larger than the

value of *V_manual* (Table 1), suggesting that the automatic method extracted an unnecessary domain when the initial contour was generated. Figure 15 shows (a) the original image of *Dunaliella parva* and (b) the initial contours generated by the automatic method (also see Figure 13).



(a) **(b)**

Figure 15. (a) The original image of *Dunaliella parva*, and (b) the initial contours (black pixels) generated by the automated method.

The initial contour extracted for the granular inner tissue domain is represented in Figure 15(b). The level set method extracts a greater region than the manual method, contributing to high *Positive_error* values (130% in *Dunaliella*; Table 1). *Negative_error* values less than 20% of *V_manual* values indicate that the proposed method adequately extracts the spherical cellular tissue. The *Negative_error* for region E is equivalent to 60% of the *V_manual* value (Table 2), probably because this region is a tubular structure.

The capacity of an active contour model to extract objects with vague contours depends on the internal energy. When the internal energy is large, a spherical contour is extracted. However, when the external energy is large, the extraction of contours in inner cell structures with low gradient intensities is problematic. Therefore, the automatic method did not sufficiently expand the contour in the case of the tubular cell tissue, and only an initial contour and the domain of the neighboring cellular tissues were extracted. For this reason, values of the *Negative_error* increased. The extraction of "other" inside tissue contours was apparently unaffected by these considerations.

6. CONCLUSIONS

We propose an automated 3D visualization for ultra-high voltage electron microscope tomograms using the level set method. The method generates an initial contour in the vicinity of manually recognized cell structures. Experimental verification confirmed that the automated method can extract the contours of most cellular tissues, with the exception of tubular-shaped structures. The method may expedite tomographic processing techniques.

Future research will explore the possibility of extracting contours for tubular-shaped tissues.

7. REFERENCES

[Cat03] Cates J.E., Lefohn A.E., Whitaker R.T. GIST: An interactive, GPU-based level set segmentation tool for 3D medical images. *Medical Image Analysis* 8, No. 3, pp. 217-231, 2003.

[Chu05] Chunming, L., Chenyang, X., Changfeng, G., Martin D, Fox. Level set evolution without re-initialization: A new variational formulation. *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)* 1, pp. 430-436, 2005.

[Joa06] Joachim, F. (ed.) Electron tomography: Methods for three-dimensional visualization of structures in the cell, Springer, Albany, New York, pp. 1-15, 2006.

[Set99] Sethian, J.A. Level set methods and fast marching methods evolving interfaces, in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge, England, Cambridge University Press, pp. 214-227, 1999.

[Sta03] Stanley, O., Nikos, P. Geometric level set methods, in Imaging, Vision, and Graphics, Springer-Verlag New York, pp. 1-20, 2003.

[Tak03] Takeshi, H., Akimbo, S., Misato, T., Hidefumi, K. Development of a liver extraction method using a level set method and its performance evaluation. *Journal of Computer Aided Diagnosis of Medical Images* 7, pp. 1-9, 2003.

[Wil87] William E.L., Harvey E.C. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics* 21, No. 4, pp. 163-169, 1987.

[Yos05] Yoshitaka, S., Hideaki, H. Segmentation of medical images using a level-set method. IEICE Technical Report 104, pp. 1-6, 2005.

Visualization and Analysis of Inverse Kinematics Algorithms Using Performance Metric Maps

Oliver Cardwell, Ramakrishnan Mukundan
Department of Computer Science and Software Engineering
University of Canterbury
Christchurch
New Zealand
orc13@student.canterbury.ac.nz, mukundan@canterbury.ac.nz

ABSTRACT

Iterative inverse kinematics (IK) algorithms are commonly used in graphics animations involving goal-directed motion of joint chains and articulated character models. A well-known algorithm is the Cyclic Coordinate Descent. For certain joint chain configurations and target positions, iterative methods can generate undesirable joint rotations. Similarly, certain target positions may require large number of iterations, or may not even be reachable. This paper presents a novel concept called performance metric maps as a tool for visualizing and analysing the performance characteristics of an iterative IK algorithm under parametric variations. The proposed method is particularly useful in determining how well an algorithm converges within a given region of the workspace. The paper presents the visualization aspects of the metric maps, and the results of comparative performance analysis of two IK algorithms.

Keywords

Inverse kinematics algorithms, Cyclic coordinate descent, Goal-directed motion, Articulated character animation, Performance metric maps

1 INTRODUCTION

Animation of articulated character models and the goal-directed motion of serial joint chains often require inverse kinematics (IK) algorithms that provide a converging solution for both joint angles and the target position [1],[6]. Cyclic Coordinate Descent (CCD) is a well-known iterative algorithm used in computer graphics and animation [3]. Even though the algorithm is conceptually simple and easy to implement, certain target positions may require a large number of iterations before an acceptable solution is obtained. Similar algorithms have been recently proposed either to improve the convergence of the solution, or to eliminate problems associated with large angle rotations [4],[5].

The performance analysis of such methods will have to take into account several factors that affect the parameterization of the joint chain in terms of angles, such as number of joints, link lengths, and joint angle constraints.

Several types of metrics can be defined to evaluate the performance of an iterative IK algorithm. Some of these are outlined in [4]. However, the pattern of variation of these metrics changes with the configuration of the joint chain. Given two target positions in the work space, it is often difficult to predict the value of the metric at an intermediate point. Metrics such as the minimum number of iterations, distance traveled by the end-effector, etc., do not have a linear relationship to changes in target positions.

This paper proposes a novel method for representing the values of performance metrics on a discretized pixel coordinate space that is mapped to the joint chain's workspace. The map not only provides an exhaustive set of values of a performance metric at all reachable points, but also gives an image-based visualization of its variation within the two-dimensional workspace. Here we assume that every joint other than the root has a

single degree-of-freedom given by a relative angle of rotation about a fixed axis. We also assume that motion to an arbitrary point in three-dimensional space can be considered as a combination of (i) a rotation of the chain about the root so that base, end-effector and the target lie on a plane, followed by (ii) a solution of the 2D IK problem for the chain configuration and target position on that plane. Thus a two-dimensional performance metric map is adequate for the analysis of most of the iterative IK algorithms used in computer animation. This paper gives a comparative analysis of iterative IK algorithms using performance metric maps, and shows the effectiveness of the method in analyzing the workspace characteristics of a given algorithm in terms of configuration dependent parameters.

The paper is organized as follows. The next section gives a general overview of IK structures that we will be dealing with in this paper. Section 3 looks at the CCD algorithm and also introduces a new IK algorithm that finds a solution where all joints are placed along a circular arc. Section 4 introduces the concept of performance metric maps. Section 5 gives a comparative analysis of the CCD and the circular algorithms and presents experimental results. Section 6 concludes the paper and outlines future research directions.

2 IK STRUCTURES

Common IK structures used in robotics and animation are articulated bodies. An articulated body is simply a list of joints linked end to end; forming a joint chain. Each joint in the chain has a length and an angle offset from its parent joint. One end of the joint chain is the base, and is fixed in some frame of reference, and the other end is the end-effector. In the case of a robotic arm the end-effector would be the manipulator or hand (Fig. 1). A joint chain can be described as a list of

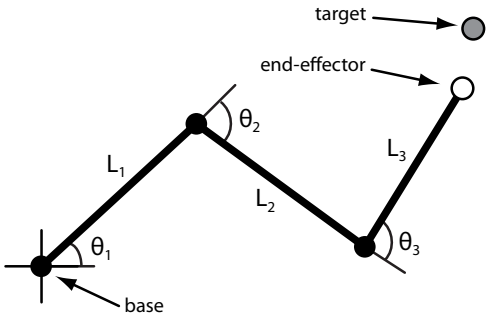


Figure 1: Joint Chain with length $n = 3$

lengths $\mathbf{l}$ and a corresponding list of angles $\mathbf{q}$ such that

$$\begin{aligned} \mathbf{l} &= [L_1, L_2, \ldots, L_n] \\ \mathbf{q} &= [\theta_1, \theta_2, \ldots, \theta_n] \end{aligned} \tag{1}$$

The state of a joint chain is given by $\mathbf{q}$. To perform forward kinematics on a joint chain, that is given $\mathbf{l}$, $\mathbf{q}$ and the base coordinates $\mathbf{b}$, find the location of the end-effector $\mathbf{e}$ then it follows that

$$\begin{bmatrix} e_x \\ e_y \end{bmatrix} = \begin{bmatrix} b_x \\ b_y \end{bmatrix} + \sum_i^n \begin{bmatrix} L_i cos\theta_i \\ L_i sin\theta_i \end{bmatrix} \tag{2}$$

However, to perform inverse kinematics on a joint chain, a function is needed that takes the desired location of the end-effector $\mathbf{e}$ and computes a valid state $\mathbf{q}$. Finding a valid state for a given joint chain configuration can be a difficult process and there are numerous methods of computing valid states.

3 ITERATIVE IK ALGORITHMS

There are a number of common parameters that are given to an inverse IK algorithm. These parameters are

n The number of joints in the chain

i The maximum number of iterations

ε The threshold distance to which the end-effector can be considered at the target location

Therefore, a typical parameter iterative IK algorithm will be passed values for n, i and ε along with the joint chain.

Cyclic Coordinate Descent

The CCD algorithm uses a heuristic approach to find a solution by iteratively rotating the links so that the end-effector moves closer to the target. Each iteration performs a sequence of rotations of links i, starting from the end-effector towards the root, trying to minimize the angle θ_i between the vector from the link joint towards the end-effector and the vector towards the target [2] (Fig. 2).

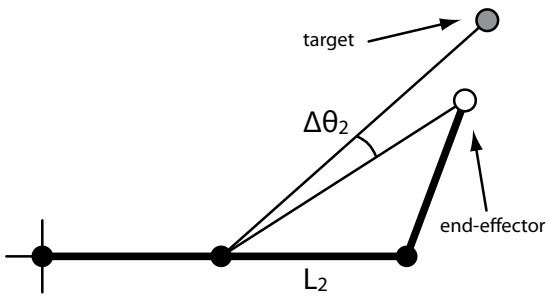


Figure 2: Joint angle rotations in a CCD algorithm

Joint angle rotations for a CCD algorithm can be easily computed and implemented in graphics applications. However, the method suffers from primarily

three types of problems: (i) certain target positions within the workspace require a large number of iterations, (ii) a solution may involve large angle rotations, and (iii) target positions near the base of the chain may result in self-intersecting configurations. Examples of these cases are shown in Fig. 3.

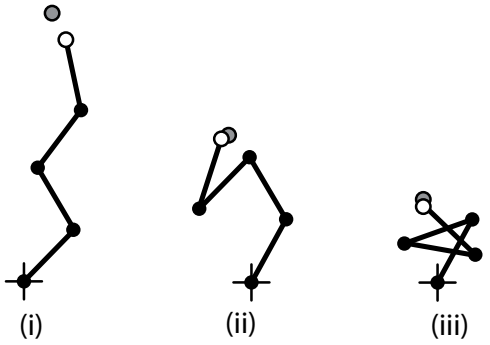


Figure 3: Limitations of the CCD algorithm

Because the CCD algorithm must visit each joint in the chain for each iteration it can be shown that the CCD algorithm has a computational complexity of $O(n)$ for each iteration.

Circular Alignment Algorithm

A few methods have been recently proposed ([4],[5]) to circumvent the limitations of the CCD algorithm. In this section, we propose a new algorithm that can be considered as a further improvement of the method proposed in [5].

If d denotes the distance of the target $\mathbf{t}$ from the base of a joint chain, then there exists a unique circumscribing circle with radius r along which all nodes can be positioned (Fig. 4).

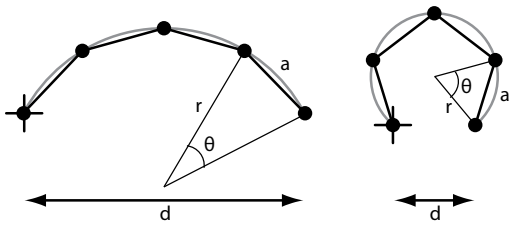


Figure 4: Circular alignment of joints

In the following, we make the assumption that all joints have the same length. If there are n joints in the chain then the chain contains $n+1$ nodes. If θ is the segment angle of a joint of length a in the circumscribing circle with radius r then it follows that

$$a = 2rsin\left(\frac{\theta}{2}\right) \tag{3}$$

And therefore we can define the relationship between lengths a and d as

$$\frac{sin\left(\frac{\theta}{2}\right)}{sin\left(\frac{n\theta}{2}\right)} = \frac{a}{d} \tag{4}$$

We seek the solution of the above equation for θ, by defining the function

$$f(\theta) = dsin\left(\frac{\theta}{2}\right) - asin\left(\frac{n\theta}{2}\right) \tag{5}$$

with the derivative

$$f'(\theta) = \frac{d}{2}cos\left(\frac{\theta}{2}\right) - \frac{na}{2}cos\left(\frac{n\theta}{2}\right) \tag{6}$$

The solution for θ is obtained using Newton-Raphson iteration

$$\theta_{i+1} = \theta_i - \frac{f(\theta_i)}{f'(\theta_i)} \tag{7}$$

with initial condition

$$\theta_0 = \frac{2\pi}{n} \tag{8}$$

This givens the initial configuration that the joint chain is curled around so the end-effector is at the base, such that the length $d = 0$. As the angle θ approaches 0 the chain opens up until d is within the threshold distance ε of the target t.

When the joints are aligned along a circular path, the joint angles will automatically assume values in an acceptable range, and there is no possibility of the chain intersecting itself. The Newton-Raphson method yields fast convergence for the parameter θ, from which the joint angles that are all equal, can be computed.

Because the Circular Alignment Algorithm (CAA) method does not need to visit each joint during an iteration it can be shown that the CAA method has a computational complexity of $O(1)$ for each iteration.

4 PERFORMANCE METRIC MAPS

Metric maps allow easy visualization and analysis of otherwise complex or dense data. A common example of metric maps are terrain or elevation maps. In these metric maps the terrain height at any point on the map is represented as a color shade. They also often include contour lines at designated elevation intervals to help us visualize the layout of the terrain represented by the map.

This basic principal can be generalized to display many other types of data. We will show how using metric maps can greatly simplify the visualization and analysis of the behavior of various iterative inverse kinematic algorithms.

In order to generate a performance metric map of the workspace of a given joint chain we need to encapsulate the whole workspace in an image. Because the workspace is a circle we need a square image and if we set the base of the chain to be the center of the image then the length of the joint chain becomes the radius. An example of this can be seen in Fig. 5.

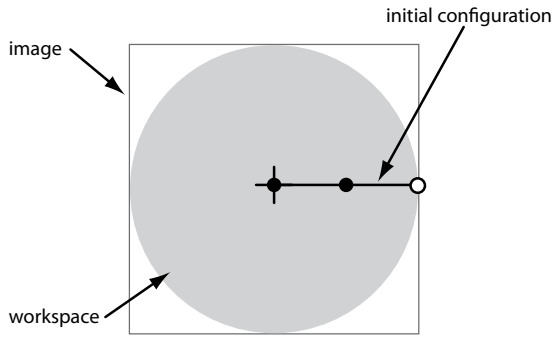


Figure 5: Joint Chain Workspace

From this initial configuration we can now iterate over each pixel in the image using the pixel's coordinates relative to the image center as the target point for the joint chain. For each of these targets we simply run an iterative inverse kinematic algorithm over the joint chain and record the desired metric, for example, the number of iterations taken to move the end-effector to within ε of each pixel can be seen in Fig. 6. Where black indicates that the algorithm failed to reach that pixel, red indicates that algorithm successfully reached the pixel on the last attempt and in hue scale down to blue which indicates that the pixel was reached in a single iteration.

It is interesting to note in Fig. 6 the dark blue (single iteration) region separates the remaining iso-surfaces into two disjoint regions. These properties and patterns cannot be analytically derived but by using metric maps these properties can be observed.

Other metrics can also be measured using the same method. An example of plotting the distance traveled by the end-effector can be seen in Fig. 7.

5 COMPARATIVE ANALYSIS

Using metric maps to help visualize the behavior of a iterative IK algorithm on a joint chain gives a much clearer picture of problem areas. In Fig. 6 it can been seen that the CCD method fails to place the end-effector at target locations in and around the initial end-effector location. The existence of this large void is somewhat unintuitive but can be clearly seen using a metric map. It is also can been seen that CCD algorithm in general requires more iterations to reach targets further from

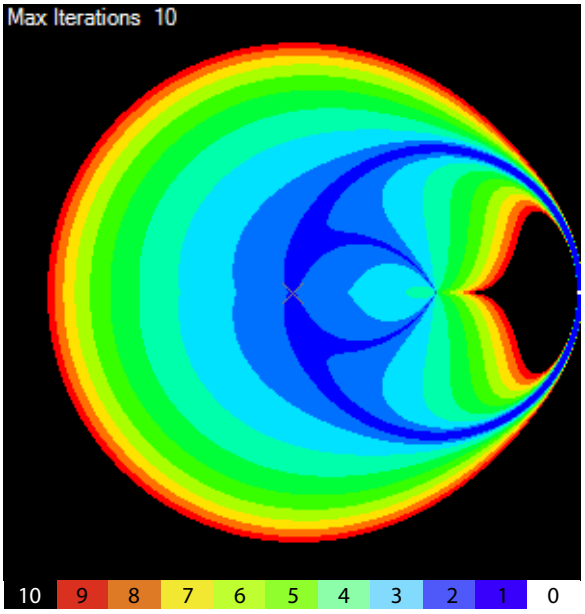


Figure 6: Metric Map of Iterations ($n = 2, i = 10, \varepsilon = 0.01$)

the base of the chain but covers most of the possible workspace.

However, as we increase the number of joints in the chain and plot their corresponding metric maps a number of interesting properties can be observed (Fig. 8). The observed structure of the metric map becomes more complex and the inclusion of more voids is evident. A interesting observation that can be made using metric maps is that the workspace coverage of the joint chain diminishes as the number of joints increases. Intuitively it could be thought that increasing the number of joints in the chain would allow the chain more freedom to move the end-effector to the target location and thus increase the workspace coverage. However it appears that adding more than three joints to a chain decreases the chain's coverage. This can been seen in Fig. 9.

An interesting optimization can be done to the workspace coverage by altering the lengths of the joints in the chain. If the lengths of the joints are set so that, starting from the end-effector, each joint is the equal to the sum of lengths before it, then the coverage is largely unaffected by the increase in the number of joints (Fig. 10). For example, if we have a joint chain where $n = 6$ then the the length ratios are $\mathbf{l} = [16, 8, 4, 2, 1, 1]$.

Using metric maps we can compare the two algorithms, CCD and CAA for iterations and workspace coverage (Fig. 12).

We can clearly see through the use of metric maps the

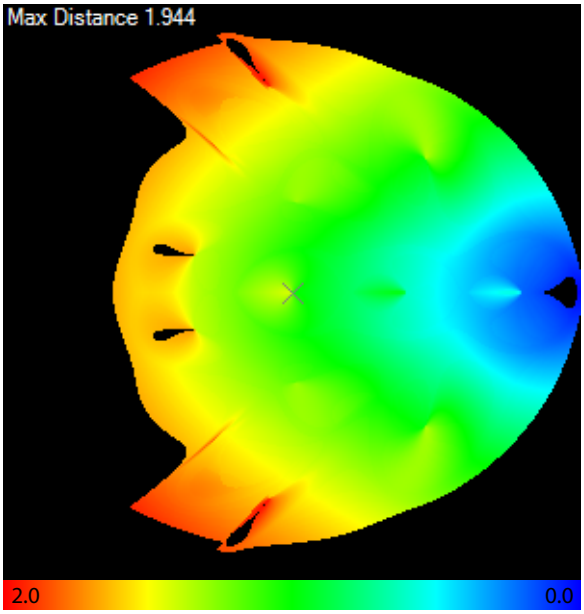


Figure 7: Metric Map of Distance Traveled ($n = 5, i = 10, \varepsilon = 0.01$)

differences in behavior between the two algorithms. In this example the CCD method has 67% workspace coverage and clearly uses up to the maximum number of iterations, colored red, to reach various target positions.

However, with exactly the same configuration we can see that the CAA method has 100% coverage and at no position needs to use up to the maximum number of iterations. In fact, the CAA method only uses a maximum of 5 iterations to reach every target point in the workspace. On closer evaluation of the CAA method we can see that the number of iterations required to completely cover the workspace is solely dependent on the size of the threshold ε. A comparison of the computational efficiency also shows that the CCD has an $O(n)$ computational complexity while the CAA operates in $O(1)$. An example of this can be seen in Figure 11.

6 CONCLUSION AND FUTURE WORK

We have shown how the use of metric maps can aid greatly in visualizing and analyzing the behavior of inverse kinematic problems. By generating metric maps for a popular iterative IK method, CCD, we were able to easily discover and identify the algorithm's various properties, including helping to formulate a new method, CAA, to address some of the negative performance aspects of the CCD method.

Although generating these metric maps can be somewhat computationally expensive, as essentially they are an exhaustive search of workspace, the generation lends itself well to parallel computational techniques. Even without parallel techniques our implementation generated 300 by 300 sample images in only a couple of seconds (see figure 11).

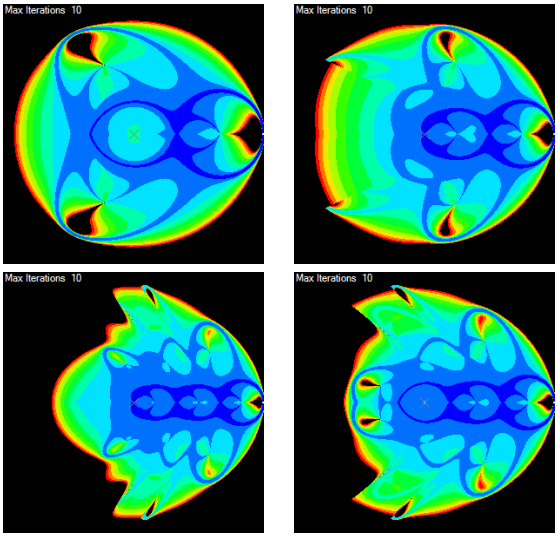


Figure 8: Clockwise from top left, $n = 3, n = 4, n = 5$ and $n = 6$

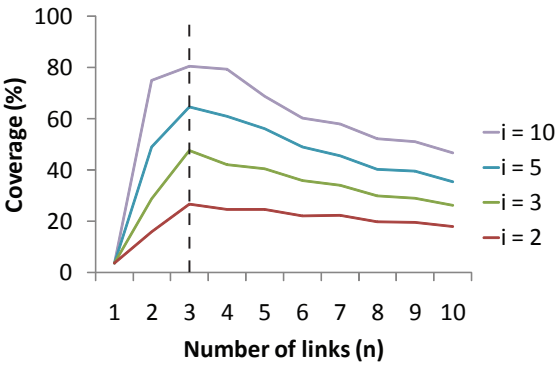


Figure 9: CCD coverage with joints of equal length ($\varepsilon = 0.01$)

The CCA method shows promise as a high performance iterative but relies on some important assumptions that may not be practical in some situations. We would like to expand the CAA method to be able to incorporate chains with joints of different lengths.

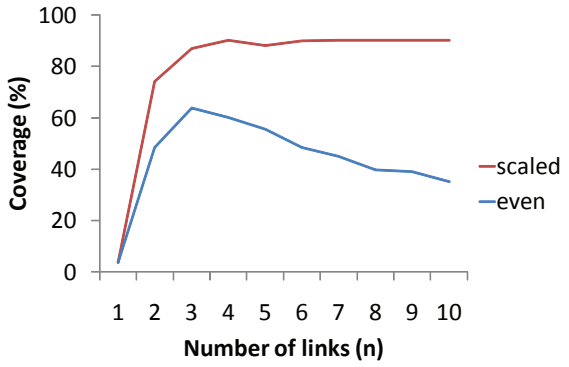


Figure 10: CCD coverage ($i = 5, \varepsilon = 0.01$)

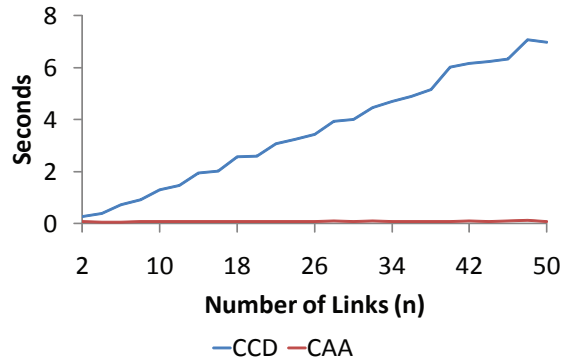


Figure 11: Performance (4x Multi-threaded) for a 300x300 metric map ($i = 20, \varepsilon = 0.01$)

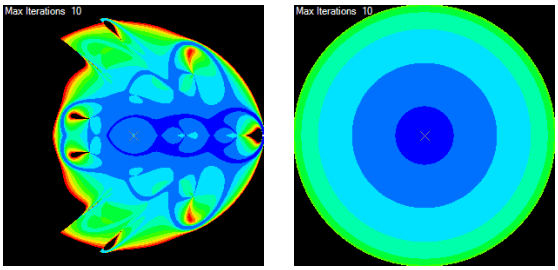


Figure 12: Comparison of CCD (left) and CAA (right) with $n = 5, i = 10, \varepsilon = 0.01$

References

[1] M. Engell-Nørregard. Inverse Kinematics The state of the art. 2007.

[2] G.Z. Grudic and P.D. Lawrence. Iterative inverse kinematics with manipulator configuration control. *IEEE Transactions on Robotics and Automation*, 9(4):476–483, 1993.

[3] J. Lander. Making kine more flexible. *Game Developer Magazine*, 11:15–22, 1998.

[4] R. Mukundan. A robust inverse kinematics algorithm for animating a joint chain. *International Journal of Computer Applications in Technology*, 34(4):303–308, 2009.

[5] R. Muller-Cajar and R. Mukundan. Triangualation-A New Algorithm for Inverse Kinematics. 2007.

[6] C. Welinan. *Inverse kinematics and geometric constraints for articulated figure manipulation*. PhD thesis, Simon Fraser University, 1993.

Detecting unstructured elements in 3D scanned scenes

M.J. González, M. Lucena, J.M. Fuertes, R. Segura, A.J. Rueda
Departamento de Informática
University of Jaén
Campus Las Lagunillas Edif. A3
23071 - Jaén, Spain
{mgmunoz, mlucena, jmf, rsegura, ajrueda}@ujaen.es

Abstract

This paper presents a technique for detecting unstructured areas in scanned 3D data. For many applications, outdoor 3D scanned data has to be filtered in order to eliminate undesirable artefacts, such as wires, vegetation, small objects, etc. Usually, this task is done manually, but it would be desirable to provide an automatic tool to reduce the preprocessing cost. The proposed technique, which consists in two stages, based on anisotropic diffusion and plane regression respectively, allows us to select most of the uninteresting data. It also has been shown good results with real data.

Keywords: Data filtering, 3D scanner, anisotropic diffusion.

1 INTRODUCTION

3D data acquisition has been turn very popular in recent times, because of the availability of affordable and very accurate scanners. A lot of 3D data is being collected from a variety of sources, outdoor and indoor, for very different purposes, ranging from reconstruction to analysis and measurement. For this reason, 3D data filtering techniques have become a very active research topic, with a variety of applications that include among others robotic vision, civil engineering, archaeology, medicine, etc.

Typical available 3D laser scanner software includes tools for processing the 3D points scanned, which constitute complex sets of data. It is often necessary to remove unwanted objects from the data (workers, equipment, temporary support structures, etc.), so a basic segmentation process is necessary. Without a priori knowledge, automated unsupervised segmentation provides unsatisfactory results [3, 1, 4]. For this reason, current 3D point-cloud management software requires manual or semi-automatic data segmentation. This can be an extremely slow and tedious operation when dealing with large complex models.

Most of the unwanted objects present in a 3D outdoors scene share common geometrical properties. In general, they do not present surface-like structures locally. We will call such objects as unstructured. Typical unstructured objects give rise to point aggregations that are: longitudinal (wires), noisy (bushes or trees branches), and very small and isolated point clusters (small objects).

In our case, we want to detect unstructured elements present in a given scene, using geometric information exclusively. Starting from a point cloud dataset, obtained from a 3D scanner, our technique allows us to detect such type of elements. By eliminating the selected points we can extract relevant structures from the scene, in order to create suitable polygonal meshes for civil engineering applications. It must be emphasized that we do not want to reduce or remove the noise present in the cloud of points, but to detect unstructured objects present into the scene.

In this paper, we propose a two-stage process for detecting such type of structures, based on anisotropic diffusion and plane regression, where 3D data are arranged as a two-dimensional matrix representing the polar coordinates of every point relative to the scanner.

This paper is organized as follows. Section 2 introduces the proposed method. Experimental results, using both synthetic and real data are shown in Section 3. Finally, Section 4 presents our conclusions and further work.

2 PROPOSED METHOD

We start by obtaining the projections of the scanned points over a range matrix I, where the column and row for a given point are determined by his horizontal and vertical angle relative to the scanner position. Each element of the matrix can be a real number, representing the distance of the corresponding point to the scanner, or be undefined if the scanner did not detect anything.

Our technique consist of two phases. The first one applies an anisotropic diffusion process to the range

matrix. This process displaces each point along the straight line that connects it with the scanner. Subtracting the resulting matrix and the original one, we can extract the high frequency component of the range matrix. One can expect that unstructured objects give rise to bigger variations in the values of the resulting matrix.

If we generate a matrix with the differences between the original range matrix and the smoothed one, we can see that not only appears big values in nonstructured zones. It also appear significant values inside the regions where the distance to the scanner changes gradually (like the ground, or a inclined wall). This is due to the anisotropic diffusion process modify the values of these regions to reach a central value. The second phase detects such variations in the subtracted matrix, measuring the difference between every actual point and a regression plane, calculated from his neighbours.

2.1 Anisotropic diffusion

Perona and Malik [7] introduced an iterative, nonlinear regularisation technique known as *anisotropic diffusion* which regularises grey-valued images while preserving important discontinuities that often contain edge information. This process can be generalised to be applied to colour [5] and disparity (range) images [6]. Our approach is based in the latter case. Being I our range matrix, we will apply this technique in order to remove the high frequencies from it, effectively displacing the points.

Anisotropic diffusion modifies the value on a point as a function of the difference with its neighbours. This difference is weighted by a conduction coefficient c. For the discrete approach of the anisotropic diffusion a four nearest neighbours discretization of the Laplacian operator can be used to update matrix values in each iteration:

$$\begin{aligned} I_{t+1} &= I_t + \lambda[c_N \cdot \nabla N(I) + c_S \cdot \nabla S(I) + \\ &\quad + c_E \cdot \nabla E(I) + c_W \cdot \nabla W(I)] \end{aligned} \tag{1}$$

where ∇ represents the gradient operator, $\lambda \in [0, 1/4]$, and the sub-indices N, S, E and W represent the North, South, East and West neighbours.

Anisotropic diffusion uses a conduction coefficient c that is 1 inside each region and 0 at the boundaries. We have used the following expression for c [7]:

$$g(x) = \frac{1}{1 + (x/K)^2} \tag{2}$$

Where x is an estimate of the range matrix gradient magnitude for the corresponding point, and K is just a threshold according to which the boundaries with contrast bigger than K will remain and the rest will tend to disappear. This value can be fixed manually or be the *noise estimator* described by Canny [2]: the accumulated histogram of the absolute values of the image gradient can be determined and the value of K is chosen that leaves 90% of the histogram values below.

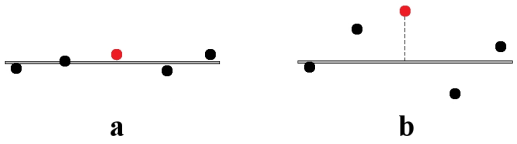


Figure 1: Unstructured point detection. a) Structured set of points, the distance between the point (red) and the ones used to estimate the regression plane (black), is small. b) Unstructured set of points: the distance between the red point and the plane is bigger.

The number of iterations can be established manually. When the process finishes, we obtain the smoothed image I'. If we compare the smoothed image I' with the original I we can see important differences in noisy zones. We will then calculate a new matrix $M = I - I'$.

2.2 Plane regression

Regions belonging to structured objects present one characteristic in M: their values can be locally adjusted to a plane with small error. We use this in a similar manner to [8], to discriminate between unstructured and structured regions. To find the best plane that adjusts the points we consider the row and column indices of the matrix as the X and Y coordinates and the values themselves as the Z coordinate. For each value inside the matrix, we can use its neighbour values to fit a plane and measure the distance between the plane and the value itself. If the value corresponds to a structured region, the distance with the plane will be very small (See Figure 1). This way we generate a new difference matrix, M', filled with the distances between every value to its corresponding regression plane, allowing us to characterize the unstructured regions.

The final selection stage will be performed by thresholding the difference matrix M' estimated in the regression phase. We use as a threshold the inflection point calculated over the histogram of M'. Isolated points are directly marked as unstructured.

3 EXPERIMENTAL RESULTS

3.1 Experimental setup

We have tested our technique on several synthetic and real world scenes. Synthetic scenes have been obtained by simulation, from a simple world composed by several simple structured objects (ground and buildings), and unstructured ones (trees, bushes and electric wires), giving us a cloud of approximately 530.000

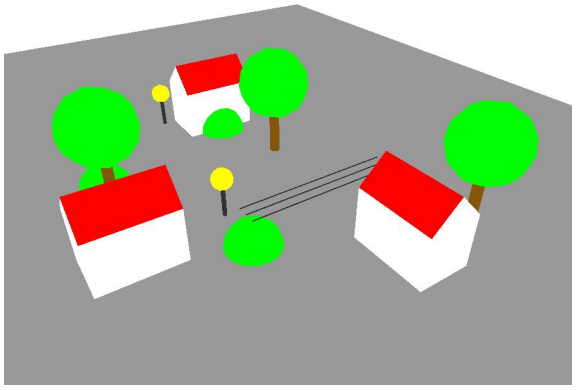


Figure 2: Synthetic scene used in experiments.

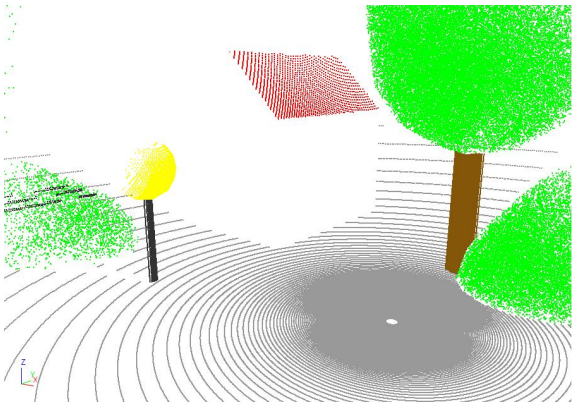


Figure 3: Synthetic cloud of points used in the experiments, generated from the scene in figure 2. showing a building, a street light, and some vegetation.

points (See Figures 2 and 3). Vegetation leaves have been simulated using randomly placed points inside a sphere.

The real world scene (Figure 4) has been obtained outdoors using a mid range laser scanner (Callidus CP 3200), showing an ancient stone bridge, surrounded by vegetation, with approximately 400.000 points. The point cloud also contains some wires, belonging to the scanning station.

We have the ground truth only for the synthetic scene, so we will show numerical results for that image only.

For the diffusion process, the following parameters have been used for all the experiments: 100 iterations, $\lambda = 0.25$, and K such that leaves an 80% of the accumulated histogram.

3.2 Results obtained

Figures 5 and 6 show the clouds of points projected over the range matrices. As we can see, there are large areas of undefined points, corresponding mainly to the sky, where the laser beam did not return to the scanning station. These points will be simply ignored in the processing stages.

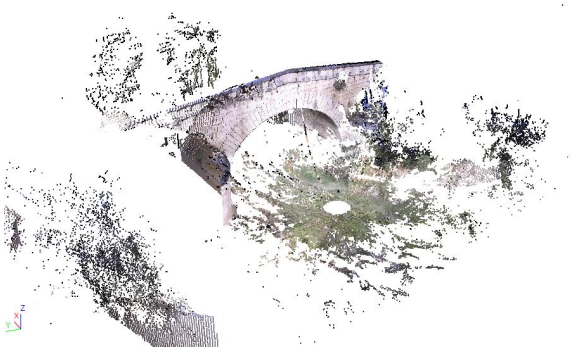


Figure 4: Cloud of points obtained outdoors by a mid range scanner.

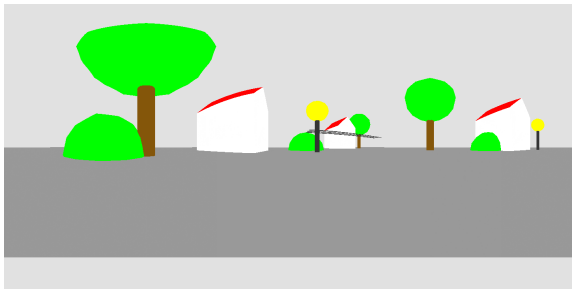


Figure 5: Projected points corresponding to the synthetic scene, using the real colours of the scene. Undefined points are shown in grey.

Figure 6: Projected points corresponding to the real scene, using the real colours of the scene. Undefined points are shown in white.

Figure 7 shows the results obtained from the synthetic cloud of points, using regression planes computed over a neighbourhood defined by a centred window size of 3. It can be seen that most of the unstructured points are correctly marked, and some of the structured points, specially those placed in high curvature areas, are marked also as unstructured. Particularly interesting is the case of the street lights, whose poles are labeled as unstructured. In fact, these objects are only slightly thicker than the electric wires.

Some numerical results are shown in Table 1. The best results have been obtained with smaller window sizes. This is due to the better tolerance to curvature in the arrangement of the points in the neighbourhoods of the structured points.

Figure 8 shows the results obtained for the real scene. It can be seen that most of the bushes are se-

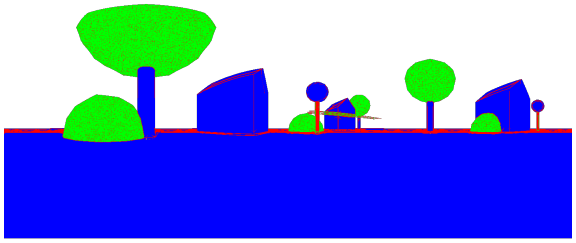


Figure 7: Results for the synthetic cloud in fake colors (blue: correct match for structured regions; green: correct match for unstructured regions; red: incorrect match). Neighbourhood window size: 3.

Window Size	Unstructured	Structured
3	92.34%	97.18%
5	92.00%	96.77%
7	89.23%	97.28%
9	89.98%	96.53%

Table 1: Accuracy levels achieved for the synthetic scene, varying the neighbourhood window size.

Figure 8: Results for the real scene. Points detected as unstructured are marked in red. Neighbourhood window size: 9.

lected correctly. The upper border of the bridge is marked also as unstructured. This is due to the presence of small bushes in this part of the bridge. We can also see that the vegetation of the ground has been not marked, because of his small height.

4 CONCLUSIONS AND FUTURE WORK

Our method can detect and mark most of the unstructured elements in outdoors 3D scenes. These unstructured elements correspond in most of the cases with undesirable objects in the scanned scene (wires, vegetation, etc.).

The first stage, composed by an anisotropic diffusion process followed by a subtraction, eliminates the low frequency components of the cloud of points, and the plane regression stage detects locally the lacking of structure. As a result, we obtain a labelling for each point, indicating the presence (absence) of local structure.

Numerical results show that the proposed method is accurate enough to give an initial estimation of the structures of interest in a cloud of points.

It is worth to mention that our method is currently used in a 3D data manipulation software, as part of a supervised point selection tool for civil engineering applications, with good results.

As a future work, we plan to test our method with other kinds of scenes, containing objects of different scales. We want also to take into account the colour information in the selection process. Finally, our method can be combined with object detection techniques to select mixed compound objects, such as trees, which have unstructured parts (leaves) and structured ones (trunk).

5 ACKNOWLEDGMENTS

This work has been partially granted by Sacyr, Junta de Andalucia, the Spanish Ministry of Education and Science, and the European Union ERDF funds under research projects 970/2007, P06-TIC-01403, P07-TIC-02773, TIN2007-67474-C03-03.

REFERENCES

[1] B. Akinci, F. Boukampa, C. Gordona, D. Huberb, C. Lyonsb, and K. Parkc. A formalism for utilization of sensor systems and integrated project models for active construction quality control. *Automation in Construction*, 15(2):124–138, 2006.

[2] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[3] A.E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(5):433–449, may. 1999.

[4] S.W. Kwon, C.T. Haas, K.A. Liapi, S.V. Sreenivasan, and J. McLaughlin. Human-assisted object fitting to sparse cloud points for rapid workspace modeling in construction automation. In *Proceedings of the 19th International Symposium for Automation and Robotics in Construction*, pages 357–362, 2002.

[5] M. Lucena, J.M. Fuertes, N. Pérez de la Blanca, and N. Ruiz. Anisotropic diffusion in colour images. In M.I. Torres and A. Sanfeliu, editors, *Pattern Recognition and Applications*, pages 81–88. IOS Press, 2000.

[6] M. Mabaar and J.P. Siebert. Smoothing disparity maps using intensity-edge guided anisotropic diffusion. In *Medical Image Understanding and Analysis 2008, 2nd-3rd July 2008, University of Dundee, Dundee, Scotland.*, 2008.

[7] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, 1990.

[8] T. Weyrich, M. Pauly, S. Heinzle, S. Scandella, and M. Gross. Post-processing of scanned 3d surface data. In *Symposium On Point-Based Graphics*, pages 85–94, 2004.

Rain Removal from Videos using the Temporal-Spatial Statistical Properties

Robin Kalia
Electronics and Telecommunications Research Institute (ETRI),
138 Gajeongdong, Yuseong-gu
305 700, Daejeon, South Korea
robinkalia@etri.re.kr

Amol Jaikar
Electronics and Telecommunications Research Institute (ETRI),
138 Gajeongdong, Yuseong-gu
305 700, Daejeon, South Korea
amol@etri.re.kr

ABSTRACT

Detection and removal of rain streaks from videos has recently become a great and challenging topic of research. This paper discusses a new technique for the removal of rain from videos using the temporal-spatial statistical properties. For this the temporal statistical properties of the pixels affected by rain are made use of, and then an efficient and easy algorithm is implemented which takes care of the effective removal of rain from videos. This technique works very well for videos with still and moving backgrounds involving moving objects with a fixed camera position. For the videos which involve the motion of the camera, the technique works well for a small rate of change of background in the camera frames. Our algorithm does not use variable and conditional parameters like the shape, size, velocity, and spatio-temporal physical model of raindrops, and camera's parameters like the aperture, focal length, and exposure time. The test results quantitatively and qualitatively illustrate that the performance of our algorithm is quite efficient in comparison to the previously existing algorithms which are state of the art techniques used for the purpose of removing rain from videos.

Keywords

Rain Detection, Rain Removal, Image Restoration, Temporal Properties, Pixel Occlusion, Spatial Filtering, Outdoor Vision and Weather.

1. INTRODUCTION

In the present day scenarios, we need to perform real time image processing and computer vision operations on real world objects. However, we have to deal with a large amount of interference and noise effects in the images of real world objects. The most important and prominent effect is that of the weather. The weather effects cause a lot of irritation to human viewers, and also affect the performance of vision algorithms for carrying out tasks like object detection, object recognition, tracking and image segmentation. Our project is based on Content Based Image Retrieval for Photo Automatic Sorting System. We have to carry out tasks like face detection, face recognition, image registration, general object detection and recognition, and key frames extraction in the images and videos of real world objects, which could vary from people to landscapes and architecture. In this case we use small features which operate on the images.

There are two kinds of outdoor weather conditions that we have to deal with: static weather conditions (fog, haze) and dynamic weather conditions (rain and snowfall). The dynamic effects of weather conditions, like the blurring and intensity altering effects of rain streaks on large portion of the images, affect the efficiency of these algorithms. Any vision algorithm which uses small features will be seriously affected by the disruptive effects of the weather conditions. Hence, we aim to reduce these disruptive effects of weather like motion blurring, and restore the image to its original form, to carry out our task of Content Based Image Retrieval with a good performance. In this paper we deal specifically with the dynamic weather conditions involving the removal of rain streaks from videos.

1.1 Related Work

Starik and Werman approached this problem by trying out temporal median filtering on pixels [Starik03a]. The problem with their developed method is that it works in the case of moderate rain conditions on clear day scenes. However, in the case of heavy rain and poor contrast scenarios, their method causes unnecessary blurring of other details in the images while retaining the blurring effects of rain considerably. Figure 1(a) shows the effect of using this approach on a scene with appropriate contrast and heavy rain conditions, and Figure 1(b) illustrates the effect of using this method on a scene with poor contrast and heavy rain conditions. It can

be seen that the image becomes a little blurred at the edges, and the rain streaks are still quite clearly evident after the temporal median filtering. Garg and Nayar have tried to remove rain using the spatial and photometric properties [Garg04a]. Their method does not work very well in the case of videos involving heavy rain. They also removed rain from videos for certain conditions by adjusting the camcorder's parameters like the exposure time and aperture [Garg05a]. However, this case does not apply very well in the case of heavy rain conditions. Zhang utilized the temporal and chromatic properties to remove rain [Zhang06a]. This method does not perform real time processing, and the chromatic property they use depends on the experimental frames. Barnum et al use blurred Gaussian to approximate a rain streak for the blurring that it causes [Barnum07a]. This can work for clear rain scenes but in the heavy rain case, a blurred Gaussian is not effectively appropriate to segment rain streaks.

Zhao and Liu implemented the histogram model to detect rain in videos [Zhao08a]. Their method uses K-means clustering, and its effectiveness is appropriate only for videos of stationary scenes taken with a fixed camera position. Park and Lee have used the Kalman filtering method to estimate the intensity of the rain affected pixels [Park08a]. This method performs real time processing of videos but it works only for a fixed camera position and still background which is not practical in real applications. Brewer and Liu model rain streaks based on the shape, velocity, and aspect ratio of rain drops [Brewer08a]. However, the aspect ratio depends heavily upon the camera's exposure time, and for a video with unknown exposure time and heavy rain, the segmentation of rain and non-rain regions is not much effective, and their algorithm does not work very well for heavy rain conditions in a video. Liu and Xu detect rain using the chromatic property, and they develop a discriminant function to eliminate false detections [Liu08a]. Their algorithm considers only videos with stationary background taken by a stationary camera, and it uses the threshold values which have to be estimated depending upon the video in consideration. They improve their method in [Liu09a] by effectively segmenting the rain and moving object pixels to work for any video with better and effective results.

1.2 Our Work

In this paper we discuss the removal of rain using the temporal-spatial statistical properties. The intensities of each pixel are analyzed for the first 15 frames. Using the statistical properties of the pixels, an algorithm is empirically developed to distinguish the rain affected pixels from the other unaffected pixels. This algorithm works for the videos of the static scenes taken by a stationary camera. To distinguish rain affected pixels in the videos with changing background taken by both stationary and moving cameras, further processing of the rain affected pixels is carried out. An empirical distinguishable property of the rain affected pixels is used. This property states that the difference in the intensities of the rain affected pixels in consecutive frames is comparatively lower than the difference in the intensities of the pixels affected by the motion of the object.

Section 2 deals with the detection and the removal of rain. Section 3 deals with the experimental results and the comparison of our algorithm with some of the previously existing algorithms. Section 4 leads to the conclusion where we discuss the benefits and the drawbacks of the proposed method. Section 5 lists the references that we have used for our study.

(a)

(b)

Figure 1: Illustrations of the result of using temporal median based filtering (a) A scene from a video with heavy rain (b) A scene from a video with heavy rain and poor contrast

2. DETECTION AND REMOVAL OF RAIN

2.1 Rain Detection

The temporal statistical properties are used for the detection of rain. We deal with the case of a stationary camera and stationary background, which may or may not contain moving objects. Then we will deal with a more general case involving the motion of camera with a changing background.

We consider the intensities of each pixel for the first 15 frames. We take the average of the intensities for each pixel over the sequential 15 frames. We have

considered a few observations reported by Garg and Nayar [Garg04a] as our initial assumptions. These observations are mentioned below.

- The intensity of a pixel shoots to a very high value as compared to its background when it is occluded by a rain drop.
- A pixel is not always covered by rain throughout the video.
- Also a pixel is almost negligibly covered by rain in more than 2 frames consecutively. The case where the pixel is covered by rain drops in more than two frames consecutively has also been accounted for by our rain removal algorithm.

So we take the average intensity of each pixel over the 15 frames. Let us say that for a particular pixel *i* in frame *n*, this specific value is $t^1_{i,n}$. The intensity values higher than this average value $t^1_{i,n}$, are considered and stored separately. Then we take the average of these higher intensities, $t^2_{i,n}$ and take the mid value between this average and the average $t^1_{i,n}$ calculated for all the frames earlier, as the threshold. Let us say that this threshold value is $t^3_{i,n}$ whose value is obtained from equation 2.1a.

$$t^3_{i,n} = \frac{t^1_{i,n} + t^2_{i,n}}{2} \tag{2.1a}$$

The intensity values greater than $t^3_{i,n}$ are empirically found out to be affected by rain, and the rest are not generally affected by rain.

We carry out this processing for all the pixels in the next 15 frames and hence forth, till the end of the video. This method will detect the rain in the case where the camera's position is fixed and the background is stationary, without involving the motion of any random object. Next we consider the case involving the motion of a random object in the video with the fixed position of the camera. This algorithm will detect the motion of a random object as false positives. In that case we can use another property of rain affected pixels which is described ahead, to distinguish them from the motion of some random object. So, we refine this algorithm further to remove the false positives.

We have observed experimentally that for a specific frame *n*, and for a particular pixel *i* affected by rain, the difference $\Delta I_{i,n,r}$ between the intensities for the frame *n* and the frame *n-1*, is lower than the difference $\Delta I_{i,n,o}$ between the intensities of the pixels affected by the motion of the random object for the frame *n* and frame *n-1*. This can be expressed mathematically with the equations.

$$\Delta I_{i,n,r} = I_{i,n,r} - I_{i,n-1,r} \tag{2.1b}$$

$$\Delta I_{i,n,o} = I_{i,n,o} - I_{i,n-1,o} \tag{2.1c}$$

$$\Delta I_{i,n,r} < \Delta I_{i,n,o} \tag{2.1d}$$

Here *r* refers to rain affected pixels and *o* refers to the pixels affected by the motion of the object. We have a general observation for these differences in intensity values.

$$\Delta I_{i,n,r} = \alpha(t^2_{i,n} - t^1_{i,n}) \tag{2.1e}$$

$$\Delta I_{i,n,o} = \beta(t^2_{i,n} - t^1_{i,n}) \tag{2.1f}$$

It is observed that the value of α lies in the range [0.2 – 0.5] and the value of β is generally greater than 0.6. This summarizes the range for α and β that we get from the videos which we used for our observation. The more precise narrow range for α and β will depend on the experimental video more accurately. Using this property of the rain affected pixels we can reduce the false positives further. This method can effectively eliminate the edges detected by the motion of the object in sequential frames, which are wrongly detected as candidate rain pixels.

Finally in the case involving the motion of the camera, we consider the videos where the background changes at very slow rate so that intensity of the pixels can be considered for the desired number of frames by their proper alignment. The limit for the frame rate of the videos, for which our algorithm seems to produce good results, lies between 10-15 frames per minute. The next section involves the removal of rain from the rain affected candidate pixels.

2.2 Removal of Rain

Here we consider the cases where a candidate pixel is affected once or more in three consecutive frames. In the case where a particular pixel *i* is affected once by rain in frame *n*, the intensity $I^{first}_{i,n}$ is calculated as an average of the intensity of the pixel in the previous frame $I_{i,n-1}$ and the next frame $I_{i,n+1}$, where the pixel is not affected by rain. This concept is taken from Garg and Nayar [Garg04a].

$$I_{i,n}^{first} = \frac{I_{i,n-1} + I_{i,n+1}}{2} \quad (2.2a)$$

If the pixel is affected by rain more than once in consecutive frames, we consider two cases. In one case the pixel is affected twice in three consecutive frames, and in the other one which is very rare in practical situations, a particular pixel is affected in all the three consecutive frames. For the former case, if a pixel *i* is affected in frame *n* and frame *n-1*, the pixel in frame *n-1* is convolved with the spatial 3×3 mask which is illustrated in Figure 2. The justification behind using this spatial filter is based upon an empirical observation which is, it is very improbable for all the pixels in the 3×3 neighborhood of the affected pixel to be covered by rain drops and especially in a streak. Also, the close neighborhood of a pixel generally has an identical intensity background pattern.

The net intensity of the pixel *i*, $I_{i,n}^{second}$ is then computed as the average of the intensity $I_{i,n+1}$ in the *n+1* frame and the intensity after spatial filtering $I_{i,n-1}^{spat.filt.}$ in the *n-1* frame.

$$I_{i,n}^{second} = \frac{I_{i,n-1}^{spat.filt.} + I_{i,n+1}}{2} \quad (2.2b)$$

We will get a similar result if the pixel in the *n+1* frame has been affected instead of the *n-1* frame.

$$I_{i,n}^{second} = \frac{I_{i,n-1} + I_{i,n+1}^{spat.filt.}}{2} \quad (2.2c)$$

Next we consider the final case, which is very rare in practical situations, where a particular pixel is affected consecutively in three frames. This is the case when the rain is very heavy as in the case of a hurricane or storm. The intensity $I_{i,n}^{third}$ of the pixel *i* in frame *n* is then calculated as the average of the intensities $I_{i,n-1}^{spat.filt.}$ and $I_{i,n+1}^{spat.filt.}$ of the pixel in the frames *n-1* and *n+1*, respectively, after it has been spatially filtered using the same 3×3 mask as shown in Figure 2.

$$I_{i,n}^{third} = \frac{I_{i,n-1}^{spat.filt.} + I_{i,n+1}^{spat.filt.}}{2} \quad (2.2d)$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 2: Spatial 3×3 mask

3. EXPERIMENTAL RESULTS

We applied this algorithm on different videos involving heavy rain, changing background, static and dynamically changing positions of camera. We consider these cases one by one to show the effectiveness of our algorithm in different practical scenarios. The results here are shown for the implementation of this technique on the standard videos used by Garg and Nayar [Garg04a], Zhang [Zhang06a], and Park and Lee [Park08a] to facilitate comparison with their methods. We compare our work with their methods since they performed completely independent, unrelated, and pioneering work in this area. The rest of the work done by other people involves the usage of some part of their algorithms to develop and modify their own technique for rain detection and its removal from videos.

These videos were taken from the work done by Garg and Nayar, and Zhang [Garg04a, Garg05a, Garg06a, Zhang06a]. We consider a simple case of a video in which rain is falling heavily in front of a brick wall causing ripples on the ground. Here the background is not changing and the position of the camera is fixed. Figure 3(a) shows the original image frame, and Figure 3(b) shows the same image frame after the application of our algorithm. It is quite clear that the rain streaks have been removed very well. Next we consider the similar case of a scene of a wall with dense rain streaks. Figure 4(a) shows an image frame from the video, and Figure 4(b) shows the same clear image frame obtained after the removal of the rain streaks. The quality of the picture obtained after the application of our algorithm is very good.

(a)

(b)

Figure 3: Video with still background and stationary camera position (a) Original image frame with clearly visible rain streaks (b) The same image frame obtained after the application of our algorithm on the video

Next we consider a more general case where the camera's position is fixed and the object is moving. Here we consider the case where there are candidate rain pixels in the foreground and the background. Figure 5(a) shows an image frame taken from a video in which a man is moving and the background as well as the camera is fixed in position. Here the rain streaks are very clear. Figure 5(b) shows the same frame after the removal of the rain streaks from the background as well as the foreground containing the moving object. The developed algorithm proves to be really effective in this case.

(a)

(b)

Figure 4: Video with still background and stationary camera position (a) Original image frame with heavy rain streaks (b) Image frame obtained after the application of our algorithm on the video containing the frame shown in figure 3(a)

(a)

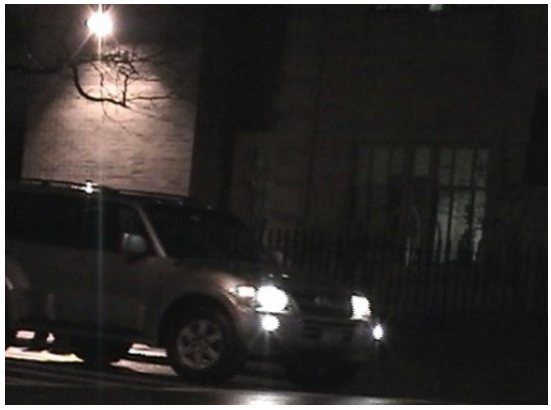
(b)

Figure 5: Video with still background and moving

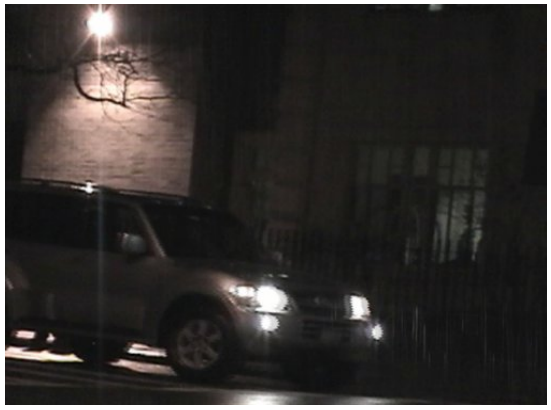
object with stationary camera position (a) The frame shows the image of a moving object with fixed background and stationary camera position containing rain streaks (b) This image shows the same frame after removal of heavy rain streaks using our algorithm

We have done qualitative comparison of our algorithm with the previously existing algorithms developed by Garg and Nayar [Garg04s], Zhang [Zhang06a], and Park and Lee [Park08a]. For this we considered a challenging case where the contrast is dark, the background is changing at a slow rate with heavy rain and the position of the camera is changing. A particular image frame is considered in Figure 6(a). Figure 6(b) shows the same frame with the rain streaks almost completely removed after the application of our algorithm. Figure 6(c) shows the frame after the application of Garg and Nayar's method. It can be seen that their method is not very effective in removing the rain streaks completely in this case. Figure 6(d) shows the image frame after the application of Zhang's method. Here the rain streaks have been removed in a better way as compared to Garg and Nayar's method. However, some rain streaks can still be perceived. Finally, Figure 6(e) shows the result after applying the Kalman filtering process as proposed by Park and Lee. Since Park and Lee's method does not perform well for videos taken by cameras with changing positions, rain streaks are still very evident in Figure 6(e) after the Kalman filtering process. The better clarity in the visual content after the removal of rain from the video using our algorithm can be compared to the results obtained after the application of other methods as shown below.

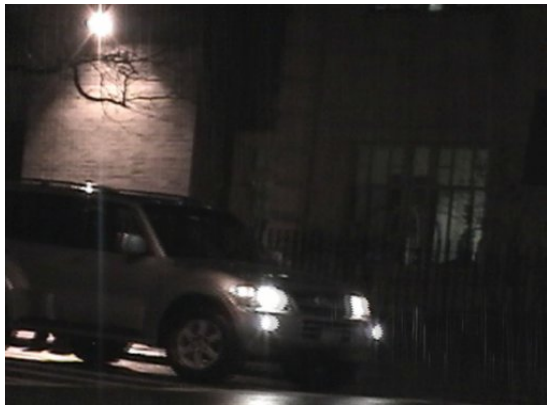
(a) Original Scene



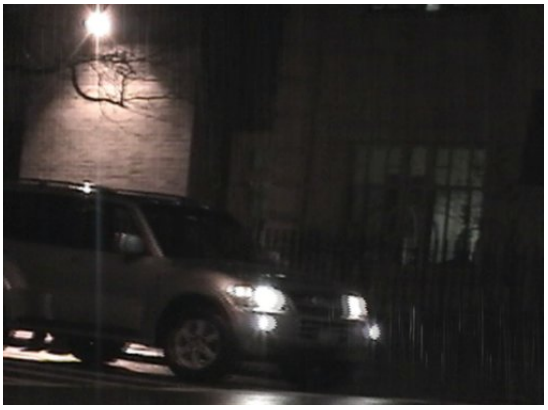
(b) Proposed Method



(c) Garg and Nayar's Method



(d) Zhang's Method



(e) Park and Lee's method

Figure 6: Qualitative comparison of our algorithm with the previously existing algorithms. Image frame is taken from a video having a dark contrast with changing background and dynamically changing camera position (a) Original scene (b) Rain streaks have been almost completely removed with the help of our algorithm (c) This image frame shows the result after the application of Garg and Nayar's method (d) Image frame after the application of Zhang's algorithm (e) This image frame shows the result after the application of Kalman filtering process as implemented by Park and Lee.

As we had mentioned in the introduction section, our purpose for this work is to restore the images which have been considerably weather degraded, to carry out tasks like object recognition, object detection, and image registration using vision algorithms. In this case we detect feature points on the image after which we carry out techniques like SIFT, SURF and MSER. It is observed that we do not get proper points in the images which have been degraded by rain. Hence, we establish the quantitative performance of our method in terms of the detection of proper feature points using the Harris-Affine Detector and the Hessian-Affine Detector [Mikolajczyk04a]. This is a completely new approach in comparison to the previous approaches to quantitatively judge the efficiency of their algorithms. Our method for evaluating quantitative performance is very specific to our objective for carrying out this work. We hope that this kind of evaluation has a potential for further research where the aim of restoring weather degraded images is to carry out content based indexing and retrieval in images and videos.

Table 1 and Table 2 illustrate the performance of Harris-Affine and Hessian-Affine Detectors on the images which have been illustrated in Figure 3, 4 and 5.

Table1: Harris-Affine Detector

Image	Resolution	Correct Number of Points Detected	Total Number of Points Detected	Time Taken
Figure 4(a)	368×288	109	550	1.2s
Figure 4(b)	368×288	114	508	1.183s
Figure 5(a)	320×304	44	239	1.033s
Figure 5(b)	320×304	47	215	1.033s
Figure 6(a)	504×376	115	767	2.333s
Figure 6(b)	504×376	139	710	2.317s

Table2: Hessian-Affine Detector

Image	Resolution	Correct Number of Points Detected	Total Number of Points Detected	Time Taken
Figure 4(a)	368×288	120	219	0.433s
Figure 4(b)	368×288	136	185	0.417s
Figure 5(a)	320×304	56	133	0.333s
Figure 5(b)	320×304	66	124	0.333s
Figure 6(a)	504×376	148	311	0.817s
Figure 6(b)	504×376	153	296	0.800s

It can be seen that the proper number of points detected using the Harris-Affine and Hessian-Affine detectors is more in the case of restored images in comparison to the weather degraded images, where the improper number of feature points detection along with the time taken for it, is more.

4. CONCLUSIONS

The experimental results show that the proposed algorithm works very well for different scenes with still and moving backgrounds with moving random objects having varying textures and still and moving camera positions. We have implemented the complete setup on MATLAB platform. The results show that the efficiency of our algorithm is comparable to the previously existing algorithms. We have dealt with the particular cases where a pixel is covered by rain in more than one frame which gives us advantage over Garg and Nayar's method where the average of the intensities of the neighboring pixels in the same frame is taken as the intensity of the pixel in that frame.

Our method has a small latency, so it could be used in real time processing applications where latency does not need to be strictly negligible. This gives us an advantage over Zhang's method which considers many frames from the complete video for processing and thus is not suitable for real time processing applications. Also, Park and Lee's method of Kalman filtering process does not apply well for videos

involving changing background and changing camera position. Thus, although their method does not involve any significant latency, it considers very specific cases which are not that practical in day to day applications. Our image quality is comparable to other methods for the general case involving slowly changing background, along with the changing camera position. Also, we do not consider the size, velocity, shape, and any physical model of rain streaks, and the external parameters like camera's aperture size, focal length, and exposure time.

On the other hand, there are a few limitations of this method as well. The small latency makes the method unsuitable for real time processing applications requiring no latency at all. Also in the case of videos having image frames with very large resolution, this method would require a lot of storage memory which may make it unsuitable for some specific practical applications. In the case of heavy rain when the spatial filter is convolved two times in two frames for a given pixel position, there is a slight degradation of quality of the video, in terms of a little blurring of the details. Still the video is better in terms of quality after the removal of heavy rain from it. The range of α and β is empirical and in some cases there are misclassifications between the rain affected and the moving object pixels. For videos with very bright background, the rain affected pixels may not be detected properly leading to insufficient removal of rain from the frames. This method cannot deal with videos with very fast changing background and camera position. Figure 7 illustrates two cases where this algorithm is not effective. In Figure 7(a), the camera's frame change rate is very fast; whereas in Figure 7(b) the rain is very heavy along with mist as experienced in a hurricane or storm. Future work focuses upon dealing with these issues.

(a) **(b)**

Figure 7: Scenes from the videos where the performance of the algorithm is not effective (a) A scene from a video where the background is changing at a high frame rate (b) A scene from a video with extremely heavy rainfall which is experienced in a hurricane or a storm.

5. REFERENCES

[Starik03a] S. Starik and M. Werman, "Simulation of Rain in Videos," in International Workshop on Texture Analysis and Synthesis, 2003.

[Mikolajczyk04a] Krystian Mikolajczyk and Cordelia Schmid, "Scale and Affine Interest Point Detectors," in International Journal of Computer Vision 60(1), pp. 63-86, 2004.

[Garg04a] K. Garg and S.K. Nayar, "Detection and Removal of Rain from Videos," in Proc. CVPR, vol. 1, pp. 528-535, 2004.

[Garg05a] K. Garg and S.K. Nayar, "When Does a Camera See Rain?," in Proc. ICCV 2005, vol. 48, no. 3.

[Garg06a] K. Garg and S.K. Nayar, "Photorealistic Rendering of Rain Streaks," in Proceedings of ACM SIGGRAPH 2006.

[Zhang06a] Xiaopeng Zhang and Teck Khim Ng, "Rain Removal in Video by combining Temporal and Chromatic Properties," in Proc. Multimedia and Expo, 2006.

[Barnum07a] Peter Barnum, Takeo Kanade, Srinivasa G Narsimhan, "Spatio-Temporal Frequency Analysis for Removing Rain and Snow from Videos," in PACV Workshop at ICCV 2007.

[Zhao08a] Xudong Zhao, Peng Liu, Jiafeng Liu, and Xialong Tang, "The Application of Histogram on Rain Detection in Video," in Proceedings of the 11th Joint Conference on Information Sciences, 2008.

[Liu08a] Liu Peng, Xu Jing, Liu Jiafeng, Tang Xianglong, Zhao Wei, "A Rain Removal Method Using Chromatic Property for Image Sequence," in Proceedings of the 11th Joint Conference on Information Sciences, 2008.

[Park08a] Wan-Joo Park and Kwae-Hi Lee, "Rain Removal using Kalman Filter in Video," in International Conference on Smart Manufacturing Application, April 9-11, 2008.

[Brewer08a] Nathan Brewer and Nianjin Liu, "Using the Shape Characteristics of Rain to Identify and Remove Rain from Video," in SSPR & SPR '08, Proceedings of the 2008 Joint IAPR International Workshop on Structural, Syntactic, and Statistical Pattern Recognition, LNCS 5342, pp. 451 – 558, 2008.

[Liu09a] Peng Liu, Jing Xu, Jiafeng Liu, Xianglong Tang, "Pixel Based Temporal Analysis Using Chromatic Property for Removing rain from Videos," in Journal of Computer and Information Science, Vol. 2, No. 1, February 2009.

Approximate importance sampling of functions reconstructed from spherical harmonics

Martin Berger
Charles University
Prague, Czech Republic
maca.berger@gmail.com

ABSTRACT

The ability to generate random samples that match a spherical PDF given in terms of spherical harmonic coefficients is very important in many fields of computer graphics. Recent work has shown that generating such samples can be done efficiently, but the published methods are not robust in the presence of reconstruction errors which manifest themselves as negative values of the PDF. In our paper, we extend the approach so that it can handle such errors, and generates uniform distribution of samples in the negative parts of the sampled function while preserving a distribution that matches the original function elsewhere. The overall distribution approximates the original function and guarantees that there are no parts of the spherical domain which remain unsampled. This property makes the scheme suitable for use in unbiased Monte Carlo rendering.

Keywords: Monte Carlo rendering, importance sampling, spherical harmonics.

1 INTRODUCTION

Spherical harmonics are a set of functions $y_l^m(\theta,\phi)$ which form a basis of square-integrable functions defined over the spherical domain. Thus, any such function can be represented as a series of coefficients in this basis. In addition, spherical harmonics have some interesting properties, such as support for rotations and convolutions, which may favor them over other similar bases. This lends to many applications in computer graphics, where functions defined over the sphere or hemisphere are very common. BRDFs ([4]), precomputed radiance transfer ([8]) and irradiance environment maps ([6]) are examples of such functions.

Recently, an efficient strategy for importance sampling of functions given as spherical harmonics coefficients has been introduced in [3]. The ability to effectively produce high quality sample distributions broadens the scope of applications of spherical harmonics to other fields of computer graphics such as unbiased Monte Carlo rendering.

Spherical harmonics are not without limitations, though. The projection of a band-unlimited function to spherical harmonics will yield an infinite sequence of non-zero coefficients, which for practical purposes needs to be truncated. This step introduces errors to the reconstructed function, which manifest themselves as the so-called ringing artifacts. Specially, for a strictly positive function f, its reconstruction $\hat{f}$ can have parts with negative values. The importance sampling scheme of Jarosz et al. is particularly sensitive to this kind of problem, because the hierarchical warping process used to generate the samples is undefined for negative values (negative values can't be used to construct a valid PDF[1]). Simple clamping of the negative values to zero will lead to bias as there will be parts of the function's domain which won't receive any samples. The authors recommend adding a positive offset to the function, but it is not clear how to find a suitable value for the offset. If the offset is set too high, it will prevent negative reconstruction issues but at the same time it will degrade the quality of the resulting distribution (it will tend towards globally uniform distribution).

On the other hand, some applications might not need a sample distribution that exactly matches the reconstructed function. During our work on unbiased Monte Carlo rendering, we faced the problem of importance sampling a local radiance estimate stored as a set of spherical harmonic coefficients. Here, the function we are trying to sample is inaccurate anyway, so an approximate sampling strategy is sufficient.

The contribution of this paper is a modification of the sampling scheme of Jarosz et al., which overcomes the reconstruction problems without function offsetting. Our method doesn't always generate samples that match the sampled function closely, but avoids bias from negative reconstruction values.

2 RELATED WORK

2.1 Background

Real spherical harmonic basis functions are defined by:

[1] Probability density function

$$y_l^m(\theta,\phi) = \begin{cases} K_l^m P_l^{|m|}(\cos\theta)\cos|m|\phi, & \text{for } m \geq 0 \\ K_l^m P_l^{|m|}(\cos\theta)\sin|m|\phi, & \text{for } m < 0, \end{cases} \quad (1)$$

where K_l^m are constants and P_l^m are the associated Legendre polynomials. For a detailed description of spherical harmonics and their properties, see [7] or [2].

Due to orthogonality, the coefficients of a function f projected onto the spherical harmonic basis can be obtained from:

$$c_l^m = \int\!\!\int_{4\pi} f(\theta,\phi) y_l^m(\theta,\phi)\, d\theta\, d\phi \quad (2)$$

For practical purposes, we truncate the series by setting $y_l^m = 0$ for $l > N$, where N is a pre-determined maximum band. During reconstruction, we approximate the original function by summing the basis functions weighted by the coefficients c_l^m:

$$f(\theta,\phi) \approx \hat{f}(\theta,\phi) = \sum_{l=0}^{N} \sum_{m=-l}^{l} c_l^m y_l^m(\theta,\phi). \quad (3)$$

Mathematically, truncated spherical harmonics expansion can be shown to be the minimizer of the least squares error functional:

$$\int_{4\pi} (f(\Omega) - \sum_{l=0}^{N} \sum_{m=-l}^{l} c_l^m y_l^m(\Omega))^2 \, \mathrm{d}\Omega. \quad (4)$$

Minimizing the square of the error allows the result to oscillate about the original function which gives rise to the so called ringing artifacts. There are a number of techniques how to reduce this effect, for example filtering the resulting coefficients, using constrained least squares projection or offsetting the function before its projection. A survey of these techniques along with a rigorous mathematical description of the problem can be found in [5], [1] and [7].

However, none of these techniques can guarantee a non-negative reconstruction $\hat{f}$ for an arbitrary non-negative function f and arbitrary maximum band N in general.

2.2 Hierarchical sampling

Here, we give a brief overview of the sampling scheme introduced in [3]. The process starts with a uniform sample distribution over the whole surface of the sphere. In the second step, we split the domain into four quadrants and compute the integrals of the function over these sub-domains. The four computed values serve as an importance function, which is used to warp to sample set. This step is then recursively repeated on the four quadrants.

Technically, the warping step is accomplished by doing a warp along the vertical axis first and then along the horizontal axis. For a domain T and its quadrants A,B,C,D (see Figure 1), this means we compute the integrals $I_1 = I_A + I_B$ and $I_2 = I_C + I_D$ of the reconstructed function and warp the set of the samples according to probabilities $p_{AB} = \frac{I_1}{I_T}$ and $1 - p_{AB}$. Warping along the horizontal axis is analogous. The effect of the warping step is that more samples are placed in areas with large values of $\hat{f}$.

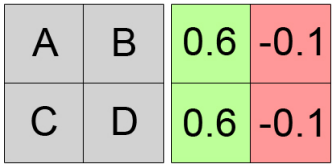


Figure 1: Left: definition of quadrants and integrals of the corresponding domains used in the text. For visualization purposes, we have mapped the spherical surface domain to a square. T denotes the union of all A, B, C, D. Right: one of the possible scenarios where some of the integrals are negative.

Warping continues in this fashion recursively up to a predefined maximum warping depth. The PDF of each sample is then computed from the ratio of the integral over the node containing the sample and the integral over the whole sphere.

This method generates samples that are distributed exactly proportionally to values of the reconstructed function $\hat{f}$ as long as the reconstruction is positive. However, once we encounter negative values for the integrals, we cannot perform the warping step and the scheme breaks. The authors propose adding a positive offset to the function before projection, but finding a suitable value for this parameter automatically is an open problem.

3 OUR APPROACH

Instead of trying to avoid negative reconstructed values completely, we use different rules during the warping process so that it can handle them in an unbiased way.

3.1 Warping step

The basic warping step is similar to [3]. First, the samples are warped along the vertical axis and then along the horizontal axis. As opposed to the original approach, we don't use the values of the integrals I_1, I_2, and corresponding probabilities $p_{AB} = \frac{I_1}{I_T}$, $p_{CD} = 1 - p_{AB}$ directly, but rather we use the values

$$\hat{p}_{AB},\ \hat{p}_{CD} = 1 - \hat{p}_{AB} \quad (5)$$

, where

$$\hat{p}_{AB} \text{ is } p_{AB} \text{ clamped to the } [\varepsilon, 1-\varepsilon] \text{ range} \quad (6)$$

for $0 < \varepsilon \leq \frac{1}{2}$. Warping along the second axis is analogous.

Our observation is that this enables us to continue warping even if some of the integrals I_A, I_B, I_C, I_D are negative, but only as long as the total integral I_T is positive. In effect, we modify the function we are trying to sample so that it has positive values of the respective integrals. If the total integral I_T is negative, we terminate the recursion immediately, which leaves the sample uniformly distributed in the domain of T as we have no suitable definition of corresponding sample distribution in this case.

Our scheme guarantees that we always get valid sample distributions and that there are no areas completely without any samples. This follows from the fact that at each warping level, the probability of each quadrant is at least ε^2, so for K levels of recursion, we have $p_X \geq \varepsilon^{2K} > 0$ for all respective sub-regions X of $\hat{f}$. This along with the fact that we can compute the PDF of each sample exactly means that the importance function is nonzero over the whole domain and the Monte Carlo estimator remains unbiased for any $\varepsilon \in (0, \frac{1}{2}]$.

3.2 Sample PDF

The PDF of each sample after the warping step can no longer be computed simply as the integral of the containing node divided by the total integral. This is because our clamping rule diverts the PDF of generated samples from the original function. Instead of the original calculation, we compute the final PDF incrementally during the recursion. Each warping step modifies the probability of a given quadrant from the original $\frac{1}{4}$ to $\hat{p}_h\hat{p}_v$ for the respective horizontal and vertical probabilities computed from $\hat{f}$. Therefore, we need to scale the sample PDF by the factor $\frac{\hat{p}_h\hat{p}_v}{\frac{1}{4}}$ for each warping level.

If we start with a PDF of a uniform distribution over the whole spherical domain, the final PDF of the sample (after k levels of warping) will be:

$$\frac{1}{4\pi}\prod_{l=1}^{k}\frac{\hat{p}_h\hat{p}_v}{\frac{1}{4}} = \frac{4^k}{4\pi}\prod_{l=1}^{k}\hat{p}_h\hat{p}_v \qquad (7)$$

3.3 The role of ε

The value of ε generally affects the uniformity of the resulting distribution.

Setting ε near zero will yield a distribution, whose PDF matches the original function very closely, but very few samples will be in the regions of negative reconstruction. In the limit case of $\varepsilon = 0$, our method will return the same sample distribution as the original method of Jarosz et al. for functions which do not exhibit negative reconstruction issues.

On the other hand, setting $\varepsilon = \frac{1}{2}$ will yield globally uniform distribution, as the probabilities will be equal in each warping step.

In our rendering system, where we sample functions that approximate local radiance estimates, we use a value of $\varepsilon = 0.01$ so that the sample distributions match the functions closely.

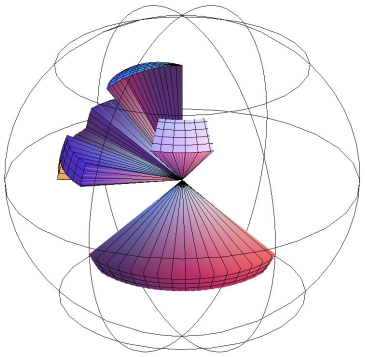


Figure 2: The original non-negative function (before projection) used for evaluation of our method. The blocky behavior and discontinuities are particularly difficult for spherical harmonics and severe ringing artifacts can be expected upon projection and reconstruction of this function.

4 RESULTS

Figure 3 shows distributions obtained with our method and with the original method from [3] with offsetting. The same number of generated samples is shown for both methods. After reconstruction, our function from Figure 2 exhibits ringing artifacts and has parts with negative values. Note that function offsetting causes the distribution to be much more uniform than the distribution from our method.

In our case, where we used the proposed method for importance sampling of local radiance estimates, the distribution generated with our method resulted in faster convergence, because fewer samples were sent to insignificant directions.

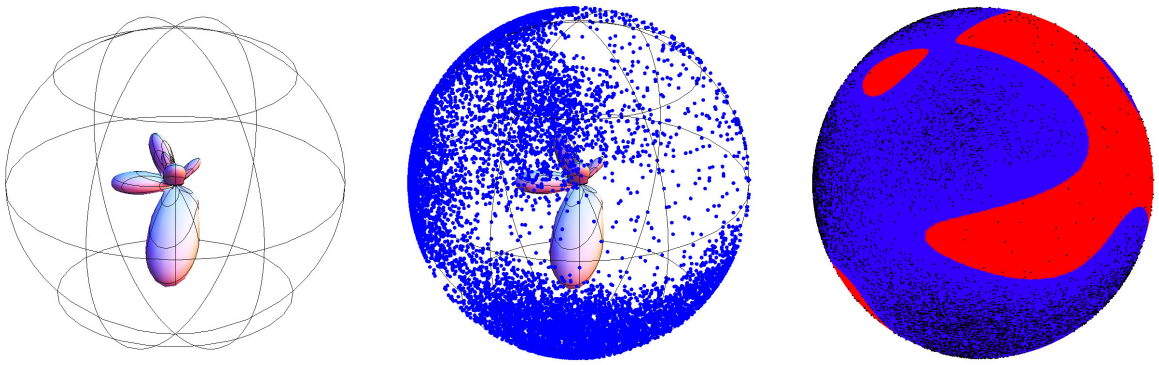
5 CONCLUSION

In our paper, we introduced a method for sampling functions given in terms of spherical harmonic coefficients, which, unlike previous methods, is robust in the presence of reconstruction errors.

The distribution generated with our method will be warped according to the sampled function in its regions of positivity, and will be uniform in its negative regions. Also, there is virtually no memory requirements or performance penalty associated with our modifications.

ACKNOWLEDGEMENTS

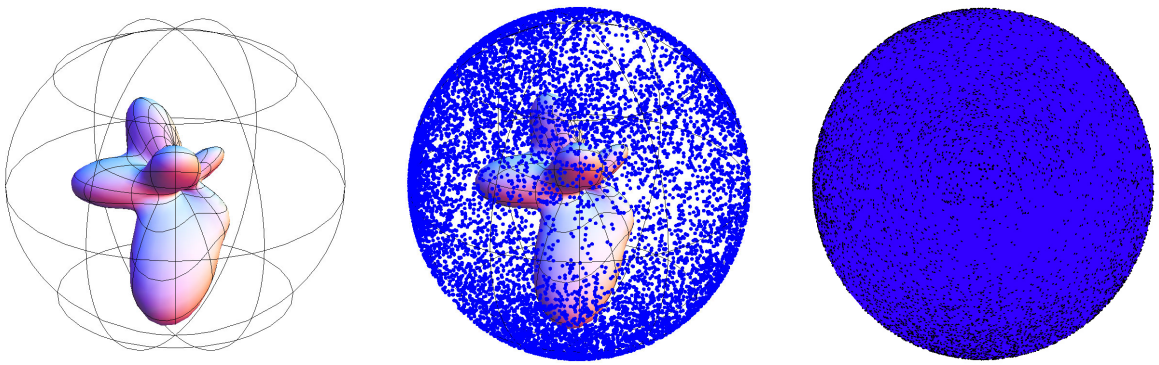
We would like to thank Alexander Wilkie, who provided valuable discussion and insights.



(a) The function reconstructed from projection to spherical harmonics using six bands.

(b) Reconstructed function along with samples generated by our scheme with $\varepsilon = 0.1$.

(c) Negative (red) and positive (blue) parts of the reconstructed function.



(d) Reconstructed function with offsetting. The minimum offset required to make the reconstruction positive across the whole spherical domain was determined by trial and error.

(e) Reconstructed function with offsetting along with samples generated by the original method of Jarosz et al.

(f) Negative (red) and positive (blue) parts of the reconstructed function.

Figure 3: A comparison of our method and the original method of Jarosz et al. The first row shows results obtained with our method. Note that the reconstructed function has large parts with negative values and that these regions do receive a fraction of the samples. On the contrary, to achieve non-negativity of the reconstructed function with the original method (the second row), a comparatively large offset value was needed, and the resulting distribution is much more uniform as a result.

REFERENCES

[1] J. P. Boyd. *Chebyshev and Fourier Spectral Methods*. Dover Publications, New York, 2001.

[2] R. Green. Spherical harmonic lighting: The gritty details. *Archives of the Game Developers Conference*, March 2003.

[3] W. Jarosz, N. A. Carr, and H. W. Jensen. Importance Sampling Spherical Harmonics. *Computer Graphics Forum (Proc. Eurographics EG'09)*, 28(2):577–586, 4 2009.

[4] J. Kautz, P. P. Sloan, and J. Snyder. Fast, arbitrary brdf shading for low-frequency lighting using spherical harmonics. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering*, pages 291–296, Aire-la-Ville, Switzerland, Switzerland, 2002. Eurographics Association.

[5] R. G. McClarren, C. D. Hauck, and R. B. Lowrie. Filtered spherical harmonics methods for transport problems. In *Proceedings of the International Conference on Mathematics and Computational Methods and Reactor Physics, American Nuclear Society*, 2009.

[6] R. Ramamoorthi and P. Hanrahan. An efficient representation for irradiance environment maps. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 497–500, New York, NY, USA, 2001. ACM.

[7] P. P. Sloan. Stupid spherical harmonics (sh) tricks. *Game Developers Conference*, February 2008.

[8] P. P. Sloan, J. Kautz, and J. Snyder. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. *ACM Trans. Graph.*, 21(3):527–536, 2002.

CUDA Expression Templates

Paul Wiemann
Computer Graphics Lab,
TU Braunschweig, Germany
p.wiemann@tu-bs.de

Stephan Wenger
Computer Graphics Lab,
TU Braunschweig, Germany
wenger@cg.tu-bs.de

Marcus Magnor
Computer Graphics Lab,
TU Braunschweig, Germany
magnor@cg.tu-bs.de

ABSTRACT

Many algorithms require vector algebra operations such as the dot product, vector norms or component-wise manipulations. Especially for large-scale vectors, the efficiency of algorithms depends on an efficient implementation of those calculations. The calculation of vector operations benefits from the continually increasing chip level parallelism on graphics hardware. Very efficient basic linear algebra libraries like CUBLAS make use of the parallelism provided by CUDA-enabled GPUs. However, existing libraries are often not intuitively to use and programmers may shy away from working with cumbersome and error-prone interfaces. In this paper we introduce an approach to simplify the usage of parallel graphics hardware for vector calculus. Our approach is based on expression templates that make it possible to obtain the performance of a hand-coded implementation while providing an intuitive and math-like syntax. We use this technique to automatically generate CUDA kernels for various vector calculations. In several performance tests our implementation shows a superior performance compared to CPU-based libraries and comparable results to a GPU-based library.

Keywords: GPU computing, parallel computing, CUDA, linear algebra

1. INTRODUCTION

In the last years general purpose computation on graphics processing units (GPGPU) has become more and more popular [Deg10, TNA+10, VKS10]. The modern GPU is not only a graphic engine but also a flexible programmable processor that can execute thousands of threads in parallel[TNA+10]. In future parallel computing will most probably get even more important. Microprocessor development will focus on adding cores rather then increasing single thread performance [OHL+08]. Since todays GPUs outclass consumer CPUs in terms of FLOPS, which is a common measure for computing capabilities, it is obvious that one should use this to speed up numerical calculations. Highly parallel linear algebra libraries like CUBLAS make use of computing power on graphics hardware but have a lack in usability. In this paper we take on this problem by introducing a technique allowing us to use a concise and math-like syntax, while utilizing the computing power of the GPU. We achieve our goal by combining CUDA and the expression templates technique.

CUDA [NVI10b] is a general purpose parallel computing architecture developed by NVIDIA. It allows one to run code, written in CUDA C, which is derived from C and has some extensions but also restrictions, on CUDA-enabled GPUs. In general, this is done by writing a so-called kernel, a function that is executed N times in N different threads. The threads are organized in warps and blocks. Each block is individually scheduled on the GPU processor cores. This means, blocks can run on any processor core in any order. Thus, a CUDA kernel scales automatically with the number of cores. A warp is a group of 32 parallel threads within a block. Each warp is scheduled individually and executes one instruction per cycle. Hence, to reach maximum efficiency the threads' execution path within a warp should not diverge, because otherwise each path will be executed separately in serial, multiplying the execution time by the number of execution paths. [NVI10b]

In our approach we use the expression template technique, which was concurrently invented by Todd Veldhuizen and David Vandevoorde in 1995 [VJ03, Vel95, IR09], to generate CUDA kernels. In general expression templates make passing expressions as function arguments in C++ possible. The expression gets inlined into the function body, preventing the overhead of callbacks. Therefore a templated class is defined, which represents an arbitrary expression. At compile time the expression gets parsed and stored as the template parameter [Vel95].

We use this technique to generate CUDA kernels, for arbitrary mathematical expressions, which then can concurrently be evaluated on the GPU.

2. RELATED WORK

Since an efficient implementation of vector algebra operations is crucial to many algorithms, there are many libraries facing this problem. On the one hand, template expression math libraries like uBLAS or blitz++[1] provide a math-like syntax but do not use the new computing capabilities on GPUs [Vel00, WK+10]. But since scientific computing through graphics hardware can be considerably faster then C-code for the CPU [CAN08], we attempt to make use of those capabilities in our approach. On the other hand libraries like CUBLAS [NVI10a] or thrust[2] are based on CUDA. While CUBLAS implements basic linear algebra functionality, thrust is a generic C++ template library for CUDA with a high-level STL-like interface. However, neither of the two provides a convenient mathematical syntax as our implementation does. Additionally, with CUBLAS, only a fixed set of functions is available, which leads to unnecessary calculations as well as to the use of temporary objects in the limited GPU memory [NVI10a]. The thrust library supports user-defined operations but requires manual specification of functor classes, which means plenty lines of code for the SAXPY operation $Y = c * X + Y$, with c as a constant. A simpler solution again requires the allocation of temporary objects. Unlike existing CUDA libraries, our implementation provides a math-like syntax and avoids temporary objects for most operators.

3. IMPLEMENTATION

Like already said we want to make the utilization of CUDA based vector-calculus easier via the expression template technique. This technique allows to pass expressions as function arguments. Those expressions get inlined, thus the code is nearly as fast as handwritten C. It can also be used to overload class operators and have the compiler generate the code to compute the result in a single pass without temporary objects. To achieve this, no subset of an arbitrary expression may be evaluated, until the entire expression is known. At compile time, the complier determines the expression type and stores it as a template parameter. Hence, all operations and operands are determined before the evaluation is evaluated and according to this the expression can be computed in one pass [VJ03, Vel95].

The CUDA compiler (nvcc) does only support a subset of C++ that includes function templates but no general template programming [NVI10b]. Thus, expression templates can not be used directly in CUDA code Instead, we use the expression templates to generate a CUDA kernel for each expression type at runtime. Those kernels are automatically executed once the execution reaches the code line, where the expression occurs.

We achieve our goal by introducing several new classes. Initially we introduce the classical classes of the expression template technique: A base class `Expression` and a vector class `cudaVec`. `Expression` represents any kind of an expression (without assignment), like $v + w$, component_wise_sin(v) or simply v. Since v is also an expression `cudaVec` is derived from `Expression`. `cudaVec` also inherits `thrust::device_vector` from the thrust library to allow for interoperability with thrust's generic interface, e.g. for reductions. Additionally, we develop the class `AssignmentExpression` with represents an assignment of an expression to a `cudaVec` and overload the assignment operator in the `cudaVec` class to instantiate an `AssignmentExpression`. For example $v = w+u$ is represented by an `AssignmentExpression`. A subset of the class structure is shown in Figure 1.

For each element-wise operation, like multiplication of a vector with a scalar or the addition of two vectors, an operator expression class is derived from `Expression`. For example, the multiplication of a vector with a scalar would be implemented in a `VectorMultipliesScalarExpression` class.

In order to allow for a concise and math-like syntax, we overload the arithmetic operators (like +, -, *, /) for expressions such that they invoke the constructor of the appropriate operator expression. An Example of an operator expression class and the associated creator function is shown in Listing 1. In the firest part the class `SumExpression` is defined which represents a plus operation with two arbitrary expressions as operands. The class has two template parameters, each of which stores an operator expression type. The template parameters' types depend on the expression types the constructor is called with. In the second part one can see the associated creator function. It is a templated function, in this case the plus operator, thus any expression can be an argument. The creator function invokes the constructor of its associated class.

In this way, we can directly write vector algebra in application code and the necessary tree of expression classes is automatically instantiated. A simple example is given in Listing 2. Since the plus operator is redefined as an creator function, it returns an instance of the `SumExpression`. Both template parameters are typed as `cudaVec`, because `a`'s and `b`'s type is `cudaVec`. Additionally the overloaded assignment operator in the cudaVec class instantiates an `AssignmentExpression` with `SumExpression<cudaVec,cudaVec>` as template Parameter. The resulting structure of the classes is shown in Figure 2.

[1] http://www.oonumerics.org/blitz/
[2] http://code.google.com/p/thrust

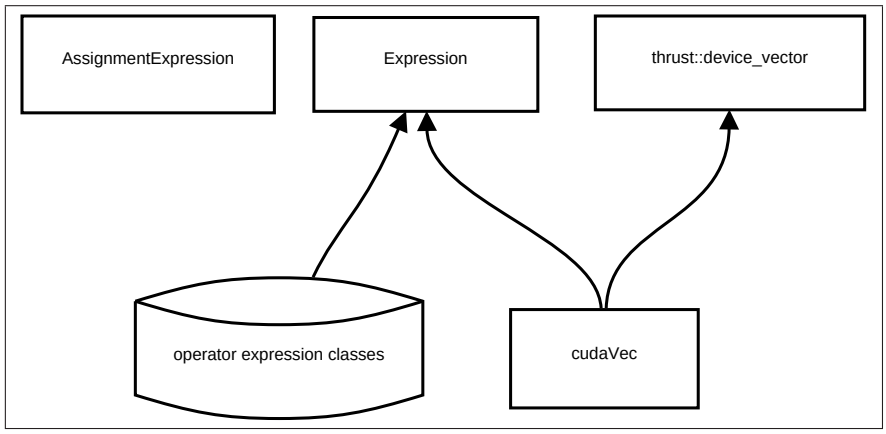


Figure 1: The class structure.

```
template <typename E1, typename E2>
class SumExpression : public Expression<SumExpression<E1, E2> > {
  public:
    const E1 &_l;
    const E2 &_r;
    SumExpression( const Expression<E1> &l, const Expression<E2> &r)
      : _l(l), _r(r) {}
//...
}

template<class E1, class E2>
SumExpression<E1, E2> inline operator +(const Expression<E1> &l,
    const Expression<E2> &r) {
  return SumExpression<E1, E2> (l, r);
}
```

Listing 1: Example for an operator expression class and the associated creator function.

```
cudaVec a(100), b(100), c(100);
//initialize a, b

c = a + b;
```

Listing 2: Sample code which instantiates the classes shown in Figure 2.

Because we want to evaluate the expressions on the GPU, each templated `AssignmentExpression` has to generate a CUDA kernel which performs the calculation. As Listing 3 shows, `AssignmentExpression` contains a static member variable to which the appropriate CUDA kernel is assigned at program startup. This is achieved by the static initializer `AssignmentExpression::init()` which is called for each templated `AssignmentExpression` and which takes care of generating the kernel code, compiling it with nvcc and loading it.

Each kernel is built according to a pattern (Listing 4). Only the parameter list and the evaluation line depend on the type of the `AssignmentExpression`. To create these two strings, we traverse the object hierarchy for the corresponding expression from the root. Each operator expression class writes its CUDA operator or CUDA function into the evaluation line and calls its parameters to do the same. Hereby the tree is gradually traversed. If a parameter is a terminal symbol (a `cudaVec` or a constant), the parameter list is extended by a new parameter and the name of the parameter is put into the evaluation line as an operand.

The example in Listing 2 generates the kernel shown in Listing 5. In this case the tree has a root typed as `SumExpression` with two children of the type `cudaVec`. As the tree is hierarchically traversed, the `SumExpression` writes its cuda operator (a +) into the evaluation line. Since both children are terminal symbols, each extends the parameter list and puts the parameter's name into the evaluation line.

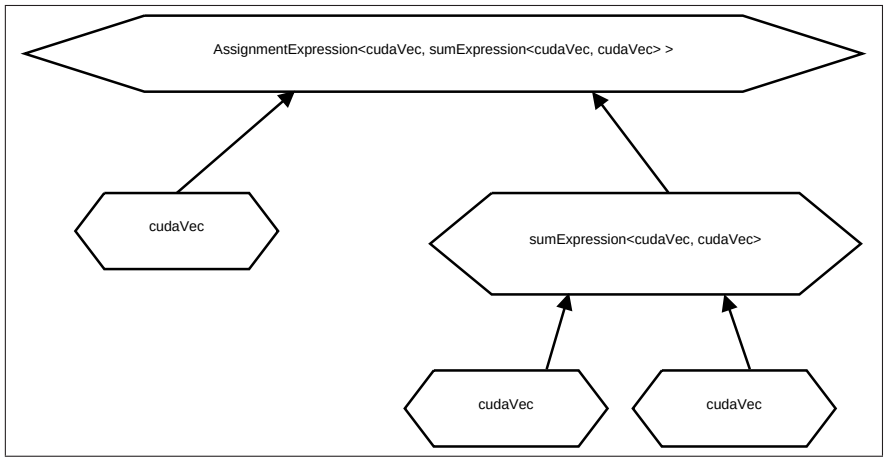


Figure 2: Hierarchy of objects that is created when Listing 1 is compiled.

```
template <class E>
class AssignmentExpression {
    //...
    static int kernelID;
    static int init();
  //...
}
template<class E>
int AssignmentExpression<E>::kernelID =
  AssignmentExpression<E>::init();
template<class E>
int AssignmentExpression<E>::init() {
  // generate, compile and load kernel
}
```

Listing 3: Implementation of the `AssignmentExpression` class.

```
extern "C" __global__ void kernel(float* a,
  /*parameterlist*/, unsigned int size){
  idx = blockDim.x * blockIdx.x + threadIdx.x;
  if (idx < size) {
    a[idx] = /*evaluation line*/;
  }
}
```

Listing 4: The Kernel prototype.

```
extern "C" __global__ void kernel(float* a, float* b,
  float* c, unsigned int size){
  idx = blockDim.x * blockIdx.x + threadIdx.x;
  if (idx < size) {
    a[idx] = b[idx] + c[idx];
  }
}
```

Listing 5: Kernel generated by compiling Listing 2.

The kernel code is compiled into an assembler-like ptx file by the CUDA compiler and then loaded with the CUDA Driver API.

Now, if an `AssignmentExpression` has to be evaluated, the expression template generated tree is traversed and each terminal symbol passes its value (in the case of `cudaVec` a pointer) to the kernel via the CUDA driver API. Thereafter the kernel is executed.

4. EXPERIMENTAL RESULTS

We now examine the experimental performance of our implementation. Zotos and Stephanides have shown that the performance of major numerical CPU-based libraries varies only by a factor of 4 [ZS]. We therefore exemplarily compare our implementation to the expression template library uBLAS, as well as to an implementation based on the thrust GPU programming framework. Our tests were performed on an Intel Core i5 750 CPU, which has four cores running at 2.67 GHz on a 64 bit Linux system with four GB RAM and an NVIDIA GeForce GTS 250 GPU with 1 GB on-board memory.

The GPU time is only reported as execution time and neither includes the time of transferring input data across the PCI express bus to the device nor the time necessary to compile the CUDA kernels. Normally, the data is transferred from the host to the device at the beginning and then all calculations concerning these data are executed. Only after the last calculation, the data is transferred back to the host. Since data transfer, which is limited by 4GB/s, can proceed while a contemporaneous kernel execution is in progress, the PCIe transfer can often be executed in the background, so that no major delay occurs [SHG09]. The kernel compiler is started only once at program startup, therefore the execution time gives us the clearest picture of the overall performance.

We perform different computations with various vector sizes to compare the efficiencies. Our first test is $a = b + c$ starting with a vector size of 1000000 up to a maximum of 50000000. Smaller sizes could not be compared since the processor time of an execution drops below 1ms. Figure 3 shows the result of this test. We gain a 55 to 58 times faster execution time for vector sizes between 1000000 and 33553920 compared to uBLAS and a little bit faster execution time compared to thrust. The execution time of our implementation increases heavily if the size goes from 33553920 to 33553921 due to the fact that we need to modify our kernel. We are forced to do so, because our first kernel computes one element per thread and now there are more elements then the maximum number of threads on the GPU used in our tests. Hence we only get a 35 times speed boost compared to uBLAS and drop slightly below the execution time of thrust. With a further increasing vector size our implementation closes the gap to thrust and increases performance compared to uBLAS as well. The described performance drop was observed in every test we took.

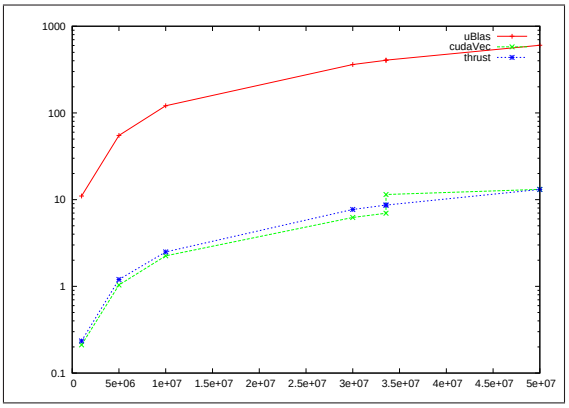


Figure 3: Test 1: a = b + c. The execution time of our CUDA implementation is 35 to 58 times faster compared to uBLAS and performs similar to thrust.

In the second and third test we test a bit more complex calculations. In comparison to uBLAS the results were similar to the first test and are shown in Figure 4 and 5. For thrust we tested both possible methods, one with a user-defined functor class and one with temporary objects. Since the memory on graphics hardware is limited the second method could only be tested for the vector-sizes up to 3355920. It is obvious that the temporary object method is up to five times slower then our implementation and also slower then the functor class approach.

In comparison to the functor class method our implementation performed faster in the second test for vector sizes below 33443920 but fell behind for greater vector sizes. Like in the first test, the gap closes with increasing size. The thrust implementation and ours performed on a similar level in the third test.

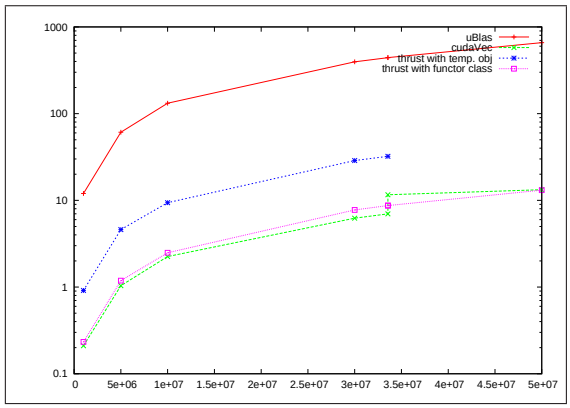


Figure 4: Test 2: a = 0.12*b + 7.54*c. The execution time of our implementation is 38 to 64 times faster compared to uBLAS and about 5 times faster then thrust with temporary objects. Thrust's implementation with a functor class and ours are on the same level.

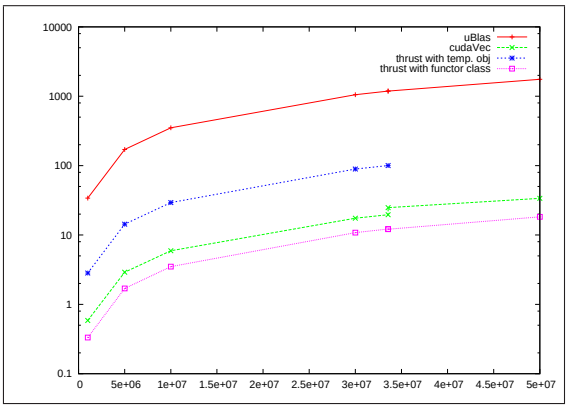


Figure 5: Test 3: $a = (b - (a + 3.75 * c) + c - 0.24 * b)/27.51 + a - 0.25 * b$. The execution time our implementation is 47 to 60 times faster compared to uBLAS and again about 5 times faster then thrust with temporary memory allocation. Our implementation and thrust's functor class approach perform similar.

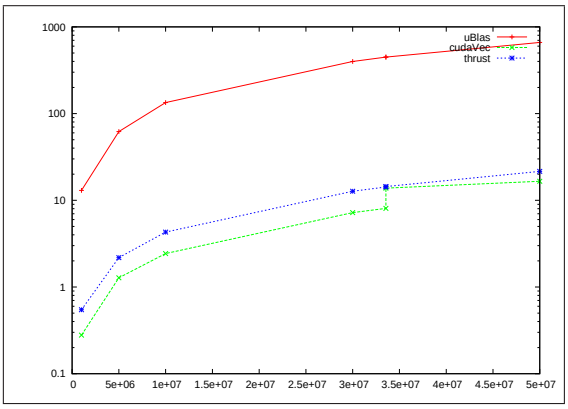


Figure 6: Test 4: $a = a/\|a\|_2$. The execution time of our CUDA implementation is 34 to 56 times faster compared to uBLAS. Thrust shows comparable results.

In a fourth test we normalized a vector. Therefor we had to compute the euclidean norm and multiply the inverse with the vector. Thrust did not provide the functionality of calculating the Euclidean norm, thus a user-defined functor class is needed. The computation of the Euclidean norm we used in our implementation is part of the CUBLAS library from NVIDIA and gets evaluated first, before the expression template generated kernel is executed. Even though we need two kernel executions there is a considerable performance gain compared to uBLAS as shown in Figure 6. The speed of our implementation and thrust's are comparable.

We also want to examine the different syntaxes of our implementation to thrust and uBLAS (Listing 6). Our implementation and uBLAS have the same concise and math-like syntax whereas thrust's syntax is more cumbersome and requires way more code.

5. CONCLUSION AND FUTURE WORK

We presented a technique, which leads to a library for vector algebra operations utilizing the capabilities of graphics hardware and still providing a math-like and concise syntax. Our implementation combines expression templates with CUDA, so that we benefit from the strengths of both.

Our experimental results show a superior performance compared to the non-GPU library uBLAS, while keeping the same brief syntax. In comparison to the thrust library our implementation performed similar to the user-defined functor class method in all tests. Compared to the thrust method with temporary objects our approach was considerably faster.

We want to point out that kernel generation occurs at run time and of course slows down the execution time of the program. But for programs with a long execution time this investment quickly pays off. There are possibilities conceivable that one could save time in this step. By caching earlier compiled kernels, recompilation can be avoided. Possible further extensions of our library include copy-free implementation of matrix algebra as well as further optimization of the kernel call parameters such as the number of threads per block.

REFERENCES

[CAN08] G. Cummins, R. Adams, and T. Newell. Scientific computation through a GPU. In *Southeastcon, 2008. IEEE*, pages 244–246. IEEE, 2008.

[Deg10] M. Degirmenci. Complex Geometric Primitive Extraction on Graphics Processing Unit. *Journal of WSCG*, pages 129–134, 2010.

[IR09] K. Iglberger and U. Rüde. The Math Library of the pe Physics Engine – Combining Smart Expression Templates and BLAS Efficiency. Technical report, Institut für Informatik, Friedrich-Alexander-Universität Erlangen-Nürnberg, 2009.

[NVI10a] NVIDIA Corporation. CUBLAS Library. `http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUBLAS_Library.pdf`, 2010.

[NVI10b] NVIDIA Corporation. NVIDIA CUDA C Programming Guide. `http://developer.download.nvidia.com/compute/cuda/3_2/toolkit/docs/CUDA_C_Programming_Guide.pdf`, 2010.

[OHL+08] J.D. Owens, M. Houston, D. Luebke, S. Green, J.E. Stone, and J.C. Phillips. GPU computing. *Proceedings of the IEEE*, 96(5):879–899, 2008.

```
/*our implementation*/
//initialize vectors a,b,c
c = 0.12*b + 7.54*b;

/*uBLAS implementation*/
//initialize vectors a,b,c
c = 0.12*b + 7.54*b;

/*thrust implementation*/
//define functor
struct functor {
  const float f1;
  const float f2;

  functor(float _f1, float _f2)
    : f1(_f1), f2(_f2) {}

  __host__ __device__
  float operator()(const float& x, const float& y) const {
    return f1*x + f2*y;
  }
};

//initialize vectors a,b,c
thrust::transform(b.begin(), b.end(),
  c.begin(), a.begin(), functor(0.12f, 7.54f)
);
```

Listing 6: A comparison of the different syntaxes from thrust, uBLAS and our implementation.

[SHG09] N. Satish, M. Harris, and M. Garland. Designing efficient sorting algorithms for manycore GPUs. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–10. IEEE, 2009.

[TNA+10] A. Tasora, D. Negrut, M. Anitescu, H. Mazhar, and T.D. Heyn. Simulation of Massive Multibody Systems using GPU Parallel Computation. In *WSCG 2010 Full Papers Proceedings*, pages 57–64, 2010.

[Vel95] T. Veldhuizen. Expression templates. *C++ Report*, 7(5):26–31, 1995.

[Vel00] T. Veldhuizen. *Advances in Software tools for scientific computing*, volume 10, chapter 2: Blitz++: The library that thinks it is a compiler, pages 57–87. Springer, 2000.

[VJ03] D. Vandevoorde and N.M. Josuttis. *C++ Templates: The Complete Guide*, chapter 18: Expression Templates. Addison-Wesley, 2003.

[VKS10] J.S.M. Vergeest, A. Kooijman, and Y. Song. Partial 3D Shape Matching Using Large Fat Tetrahedrons. *Journal of WSCG*, pages 41–48, 2010.

[WK+10] J. Walter, M. Koch, et al. uBLAS, Boost C++ software library. `http://www.boost.org/doc/libs/1_44_0/libs/numeric/ublas/doc/index.htm`, August 2010.

[ZS] K. Zotos and G. Stephanides. Analysis of Object-Oriented Numerical Libraries. Technical report, Deptartment of Applied Informatics, University of Macedonia.

When It Makes Sense to Use Uniform Grids for Ray Tracing

Michal Hapala
Czech Technical University in Prague
Faculty of Electrical Engineering
Czech Republic
hapalmic{@}fel.cvut.cz

Ondřej Karlík
Czech Technical University in Prague
Faculty of Electrical Engineering
Czech Republic
karliond{@}fel.cvut.cz

Vlastimil Havran
Czech Technical University in Prague
Faculty of Electrical Engineering
Czech Republic
havran{@}fel.cvut.cz

ABSTRACT

Commonly used hierarchical data structures such as bounding volume hierarchies and kd-trees have rather high build times, which can be a bottleneck for applications rebuilding or updating the acceleration structure required by data changes. On the other hand uniform grids can be built almost instantly in linear time, however, they can suffer from severe performance penalty, in particular in scenes with non-uniformly populated geometry. We improve on performance using a two-step approach that combines both approaches: first we build a uniform grid and test its performance. Second, using an estimate on the number of rays to be queried we either continue using the grid or build a hierarchical data structure instead. This way we select a more efficient data structure given a particular implementation of the algorithms which yields with high probability an overall smaller computational time. We evaluate the properties of this method for a set of 28 scenes.

Keywords: ray tracing, ray casting, uniform grid, kd-tree, hierarchical data structures.

1 INTRODUCTION

Ray tracing is a technique that can be used for generating images by shooting rays into a 3D scene and finding closest intersections among rays and the scene objects. This basic visibility computation is used in a core of many rendering algorithms. Although ray tracing has been known for over last four decades [App68, Gla89], it is still considered relatively slow to be massively used in real-time applications particularly for animated scenes.

Different data structures have been proposed, each one with its own advantages and disadvantages. Most commonly used are hierarchical data structures, e.g. a kd-tree [Ben75] and a bounding volume hierarchy (BVH) [Kay86]. Their main advantage is their capability to adapt to the distribution of geometric primitives – they can deal with non-uniform geometry distribution, including so-called "teapot in a stadium" type of scenes. Because of that they perform well in a vast majority of scenes encountered in real-life use. They are typically built in $O(N \log N)$ or $O(N \log^2 N)$ time using the *surface area heuristic* (SAH) [Wal06a]. Super-linear time complexity of their build algorithms can be a bottleneck in particular when tracing rays in applications with real-time requirements.

Another type of an acceleration structure is a uniform grid [Fuj86]. In its simple form it divides a scene regularly and non-adaptively into equally-sized voxels and sets the primitive references to each cell overlapped by the primitives. Ray then traverses cells along the ray path and only geometric primitives in these cells are tested for an intersection. Although this data structure can perform well in certain types of scenes (typically with uniformly distributed primitives), its performance degrades drastically in scenes with a non-uniform distribution. Despite this fact, uniform grids can still be advantageous for usage in real-time applications because their build algorithm has only a linear time complexity.

The availability of two approaches with different properties presents us with a choice whether to use either an adaptive data structures that will most likely be efficient for shooting rays, but have higher time complexity for building, or a simple regular one like the aforementioned uniform grids, which have lower build time but can have a severe performance penalty.

In this paper we propose and study such an algorithm that uses the estimate of performance properties choosing acceleration data structures on the fly. The algorithm uses a calibration phase which requires a set of scenes of different properties such as number of geometric primitives and their spatial distribution. The calibration phase is executed only once before the al-

gorithm is used for an application on a particular hardware. During the calibration phase we measure the implementation and hardware constants for building up data structures and also a practical efficiency of shooting rays. Given an unknown scene we build a uniform grid first in a short time and test its performance by sampling a small set of representative rays. Using the data from this test and data from the calibration phase we estimate if it is more advantageous to use the uniform grid or to discard it and build a hierarchical data structure instead. To make a correct decision we need to know at least roughly the number of rays to be shot in the application.

This paper is further structured as follows. Section 2 describes the previous work on the most relevant data structures. Section 3 describes the proposal of our algorithm. Section 4 shows the results obtained from the set of 28 scenes. Section 5 concludes the paper with some prospectives for future work.

2 PREVIOUS WORK

In this section we briefly recall the most important work on uniform grids and hierarchical data structures. As the number of papers is huge, we select only the most recent and important work to our approach.

Uniform Grids. The uniform grids, also called regular subdivision, were proposed by Fujimoto et al. [Fuj86]. Cleary and Wyvill [Cle88] analyse the properties of ray tracing with uniform grids in dependence on its resolution. They study the performance when the number of cells in the uniform grid is proportional to the number of objects. Other methods were also studied by Ize et al. [Ize07]. The performance of ray tracing with uniform grids has two important factors. First, the initial setup time given a ray is relatively high. Second, when the distribution of primitives in a scene is highly uniform, it is likely that the ray will stop its traversal after only a few traversal steps. Therefore, the uniform grids are for such types of scenes even more efficient than hierarchical data structures that require initial traversal phase to a first leaf. However, for moderately to highly non-uniform distribution of geometric primitives in space the uniform grids are rather inefficient as studied for example by Havran et al. [Hav00b].

Given an arbitrary ray the number of traversed cells is of order $O(\sqrt[3]{N})$ in the worst case, where N is the number of object primitives hence the number of all grid cells. The second important property is the time needed for building a uniform grid, which is only $O(N)$ provided each geometric primitive is assigned to only a constant number of cells. Wald et al. [Wal06b] studied the coherent traversal algorithm for primary rays that reaches real-time framerates. Recently, Kalojanov and Slusallek [Kal09] presented the algorithm for parallel building of uniform grids on a GPU.

Hierarchical Data Structures. Hierarchical data structures for ray tracing were studied in a number of papers. Chang [Cha04], Wald [Wal04], and Havran [Hav00a] provide the survey on the spatial data structures for ray tracing static scenes with the focus on the hierarchical data structures. The common property of the data structures is that the time complexity for building is $O(N \log N)$ since it corresponds to sorting in 3D space. While the time complexity for building is higher than the one for uniform grids $O(N)$, the time needed for ray query can be estimated by $O(\log N)$. The performance of the ray is hence much less dependent on the number of objects than for uniform grids as we also show further in the paper. The properties and relations between these data structures in general were discussed by Havran [Hav07]. Another study that compares the performances of a grid and a kd-tree was presented by Szirmay-Kalos et al. [SKH02].

Selection Algorithm. We are aware of only two algorithmic proposals that considers the use of different data structures. The first one proposed by Havran et al. [Hav00b] is based on statistical properties for the same input data. They analyse the distribution of objects in the scene and if selected statistical characteristics are low without giving any threshold, they suggest to use uniform grids, otherwise kd-trees or other hierarchical data structures such as adaptive grids. The statistics measures that are easy to compute from only the distribution of geometry in the scene are sparseness, maximum number of primitives referenced in the cell, and statistical moments as mean, variance (hence also standard deviation), skewness, and kurtosis. We recall below the formulas for these measures computed over t cells of a grid taking into account the references of geometric primitives in the cells denoted by N_i for the i-th cell.

Scene sparseness is the ratio of empty cells to all cells:

$$sparseness = \frac{\#empty\ cells}{N} \tag{1}$$

High values of sparseness could indicate inferior grid performance, as adaptive data structures generally deal better with cutting off empty space. Big gaps with no geometry in the scene create many empty cells in the grid which have to be traversed unnecessarily.

Maximum number of primitives in any cell is defined simply as:

$$maxRefs = \max(N_i), i \in 1 \ldots t \tag{2}$$

High value could be useful in detecting a "teapot in a stadium" type of scene.

Mean represents an average number of references per cell:

$$mean = \frac{1}{t} \sum_{i=1}^{t} N_i \tag{3}$$

Because the grid is built to have the number of cells proportional to the number of geometric primitives in the scene, high mean values indicate low performance as that means that there are many primitives overlapping multiple cells.

Variance gives us information about how much values differ from the mean value. It is computed using this formula:

$$variance = \frac{1}{t-1}\sum_{i=1}^{t}(N_i - mean)^2 \quad (4)$$

Higher variance corresponds to the higher differences in the data: there may be many empty cells but also many cells with high number of primitives. Standard deviation σ is computed from variance as $\sigma = \sqrt{variance}$.

Skewness describes asymmetry of the distribution and *kurtosis* describes "peakedness" of the distribution belong to higher statistical moments and are defined as:

$$skewness = \frac{1}{t}\sum_{i=1}^{t}\left(\frac{N_i - mean}{\sigma}\right)^3 \quad (5)$$

$$kurtosis = \frac{1}{t}\sum_{i=1}^{t}\left(\frac{N_i - mean}{\sigma}\right)^4 - 3 \quad (6)$$

The aforementioned metrics can be computed directly from the uniform grid based on a voxelisation approach proposed by Klimaszewski [Kli94].

Another approach to selecting a better acceleration structure on the fly was proposed by Müller and Fellner [MF99]. They create a bounding volume hierarchy for a given scene and try to find regions (nodes) that contain uniformly distributed objects. A uniform subdivision of space to a predetermined number of voxels is then created in these regions.

3 ALGORITHM OUTLINE

In this section we present the algorithm that given a scene suggest to use either the uniform grids or a hierarchical data structure in the dependence on the number of rays. We analyse such case and suggest an algorithm that estimates if it is more convenient to use an already built grid or to build up a hierarchical data structure. Our decision algorithm can be used for virtually any application of ray tracing that implements grids and hierarchical data structure in the framework. The data from the implementation are extracted in the *calibration phase* that is executed only once on a given hardware/implementation on a set of scenes.

We verified the observation by Havran et al. [Hav00b], if the suggested selection algorithm between grids and kd-trees is valid for another set of scenes. We have found out that on our set of scenes (see Figure 3) there is no scene with such low standard deviation, skewness, and kurtosis as in the study that could justify the use of uniform grids based only on the scene statistics. However, when analysing the statistical characteristics in [Hav00b] it appears that the threshold for standard deviation should be very low, such as 2.0, to justify the use of uniform grids. This leads to the higher performance of ray shooting irrespective to the number of rays even if we ignore the time needed to build the data structure.

In this paper we study another case when the number of rays to be shot is known or well estimated in advance and we account for the time needed to build the data structure. The algorithm requires the calibration phase over all s scenes in a set S_{cal} (i.e. $s = |S_{cal}|$).

The calibration phase computed for i-th scene having $N(i)$ geometric primitives for all the scenes in the set S_{cal} has four steps:

1. Build a uniform grid [Fuj86] over $N(i)$ geometric primitives of the i-th scene with the number of cells proportional to $N(i)$. Measure the time $T_B^G(i)$ to build the uniform grid.
2. Measure the time $T_R^G(i)$ for $M(i)$ ray queries using the ray traversal algorithm over the uniform grid.
3. Build a hierarchical data structure over $N(i)$ geometric primitives. Measure the time $T_B^H(i)$ needed for the build.
4. Measure the time $T_R^H(i)$ for $M(i)$ ray queries (the same ray queries as for uniform grid) using the ray traversal algorithm over the hierarchical data structure.

The data from the calibration phase are then used for an application scenario given an unknown scene S with N geometric primitives. To improve on the performance we decide on both cases assuming the knowledge or the rough estimate for the number of rays R to be queried.

Decision algorithm:

1. Build a uniform grid [Fuj86] over N geometric primitives of scene S with the number of cells proportional to N. Measure the time $T_B^G(i)$ to build the uniform grid.
2. Estimate the time t_R^G needed for computing a single ray with the uniform grid. This is carried out by sampling using a small set of rays.
3. Estimate the time T_B^H to build a hierarchical data structure from the calibration phase and from N.
4. Estimate the time t_R^H to ray trace a single ray from the calibration phase and from N.
5. If $t_R^H \geq t_R^G$ then use the uniform grid to shoot all the rays. Finish.

6. Estimate the critical point, it is the number of rays R_C, when the uniform grid and hierarchical data structure yields the same computation time taking into the account the estimated build time of the hierarchical data structure. This is computed as: $R_C = T_B^H/(t_R^G \cdot (1+\varepsilon) - t_R^H)$. The parameter ε is used only to avoid division almost by zero, we use the value such as $\varepsilon = 0.01$.

7. If $R_C \leq R$ (R is the number of rays to be queried), use uniform grids for the rest of the computation. Finish.

8. Otherwise, discard the uniform grid and build the hierarchical data structure. Shoot all the remaining rays using hierarchical data structure. Finish.

To put it short the decision algorithm above simply computes the estimate whether or not it is more advantageous to use an already built uniform grid or if it pays off to build a hierarchical data structure.

The only information computed after building the uniform grid is the build time T_B. To estimate the critical point for the number of rays R_C we need to estimate the average time t_R^G to shoot a single ray using the uniform grid, the time needed to build the hierarchical data structure T_B^H, and the time t_R^H for shooting a single ray using this (unbuilt) data structure.

Below we describe how to estimate these qualitative performance characteristics. The average time t_R^G which gives an average time to shoot a single ray in uniform grid is estimated by sampling of small number of rays such as 100 to 1000 rays. This provides an accurate estimate, the only condition is that the sampling rays represent the distribution of all rays.

The time needed to build the hierarchical data structure T_B^H is estimated using the time complexity of a build $O(N \log N)$ and from the times $T_B^H(i)$ needed to build these data structure in the calibration phase as follows:

$$T_B^H = N \cdot \log_2 N \cdot \frac{1}{s} \sum_{i=1}^{s} \frac{T_B^H(i)}{N(i) \cdot \log_2 N(i)} \qquad (7)$$

Similarly, we can estimate the time to shoot a single ray T_R^H in a hierarchical data structure under the assumption of $O(\log_2 N)$ time complexity for this operation, using the time $T_R^H(i)$ needed for the same algorithm from the calibration phase as follows:

$$t_R^H = \log_2 N \cdot \frac{1}{s} \sum_{i=1}^{s} \frac{T_R^H(i)}{M(i) \cdot \log_2 N(i)} \qquad (8)$$

Analysis and Discussion

After building the grid the algorithm above provides three possible outcomes. First, if the estimated time t_R^G to shoot a ray in the grid is lower than the estimated time t_R^H to shoot a ray in the hierarchical data structure, it does not make sense to build a hierarchical data structure. Second, for the number of rays to be shot in the range between 0 and R_C it does not pay off to build up a hierarchical data structure. This is because the time to build a hierarchical data structure is relatively high even if it provides faster processing of a single ray and for relatively small number of rays it does not pay off. Third, for the number of rays larger than R_C it is then always more efficient to discard the uniform grid, build the hierarchical data structure and use it to shoot the rays.

Below we compare a proposed algorithm combining uniform grids and hierarchical data structures with a single use of the either two data structures. When we compare it to the use of only the grids, the proposed algorithm is more efficient as it is always of the same performance or provides the speedup in cases when we detect that the grids are inefficient.

We also compare the proposed algorithm to the use of only the hierarchical data structure as we need the additional time to build the uniform grid. Favourably, the time complexity $O(N)$ is asymptotically smaller than the time complexity needed to build the hierarchical data structure $O(N \log_2 N)$. Theoretically, the time complexity needed to build the uniform grid, which is possibly later discarded, gives the time complexity increase from $O(N \log_2 N)$ to $O((N(1+\log_2 N))$ that presents the slowdown of building of only a hierarchical data structure $1 + 1/\log_2 N$. In practice when we take into account the particular implementation, the constants behind the time complexities for building them are even higher for hierarchical data structure when compared to uniform grids. Therefore such slowdown is negligible. For the test scenes used in this paper the slowdown is only 4.5% on average, with a minimum value of 2.1% (1,070,671 triangles) and a maximum value of 15.4% (528 triangles).

We can also express the maximum theoretical speedup for using the combined solution when using only the hierarchical data structure. This is only for a small number of rays with a limit of $O(\log_2 N)$ and the speedup is then $\frac{T_B^H + t_R^H \cdot \log_2 N}{T_B^G + t_R^G \cdot \log_2 N}$. The speedup reaches the average value of 28.07 with a minimum of 6.5 (528 triangles) and a maximum of 46.50 (1,070,671 triangles). We avoid the discussion for a trival case – it does not pay off to build up any hierarchical data structure if the number of rays to be shot is smaller than $O(\log_2 N)$.

4 RESULTS

We have implemented a path tracer application in C++. We report here the results for a PC equipped with Intel Core 2 Duo E4300 1.8 G Hz (Allendale) and 6 GBytes of RAM, running Windows 7 operating system in Microsoft Visual C++ 2008. For testing we have used a set

of 28 scenes, 20 of them unique and 4 scenes tessellated to triangles in two level of details, see Table 1.

For each scene we have measured uniform grid and hierarchical data structure build times and traversal times for two types of ray generation schemes. The first scheme uses rays generated from two points randomly generated on a bounding sphere of the scene, the second one uses rays generated by the path tracer. As a hierarchical data structure we used an implementation of a kd-tree. From the measurements we have computed exactly the critical point for the number of rays R_C and each scene where rendering using the uniform grid is faster according to the equations provided in the previous section. To test the quality of our estimate algorithm we have compared the exactly computed R_C and its estimated value R_{est} when the hierarchical data structure was not build. We report by how many percent the estimate of R_C is inaccurate (relative error $Err = 100 \cdot \frac{R_{est}-R_C}{R_C}$).

This was carried out for random combinations of calibration and estimated scenes as follows: always a certain number of scenes from the set C are used in the calibration stage, and the rest is used to test the accuracy of the estimate. This number C is increased from 1 to 27, thus for the first case one random scene would be the base for the calibration and twenty seven would be estimated and for the last one the situation is reversed. To gain some convergent data we have repeated the computation 5000 times for every of these cases. Both graphs in Figure 1 and Figure 2 show the average value of estimated relative errors in percent ($\frac{1}{5000}\sum Err$) in red and the average of absolute values of relative errors in percent ($\frac{1}{5000}\sum |Err|$) in blue colour.

For randomly generated rays (see Figure 1) the estimate is about 25 percent more optimistic about the quality of the grid and is quite stable in this prediction except for the extremes where there are either not enough calibration scenes or not enough estimated scenes. The estimate predicts that it is safe to shoot more rays with the grid still being faster than in reality.

For path traced rays (see Figure 2) the estimate is off by around 30 percent, but this prediction is not as stable as for the randomly generated rays and the tendency is to predict that the grid is worse than it really is. Since a big part of our path traced rays are primary rays or shadow rays, this is not an efficient sampling of the space of possible rays with regard to providing good calibration for other scenes. This can also occur for non-diffuse scenes, where glossy reflections will result in a non-uniform sampling.

From the results we see that the range of rays where its does not pay off to build the hierarchical data structure can be significant in particular for scenes with a higher number of geometric primitives. For example for the scene *phone-high* the critical point for the number of rays R_C is 2.7×10^6 rays to justify the use of hierarchical data structure for random rays and 856×10^3 rays for path tracing. The results for 28 scenes also show that without computing the estimate the selection cannot be made in general.

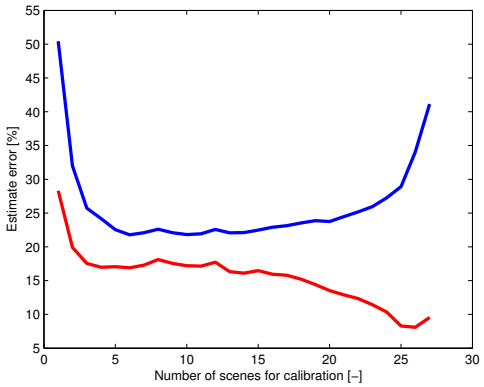


Figure 1: Estimate error for random rays calibrated on random rays.

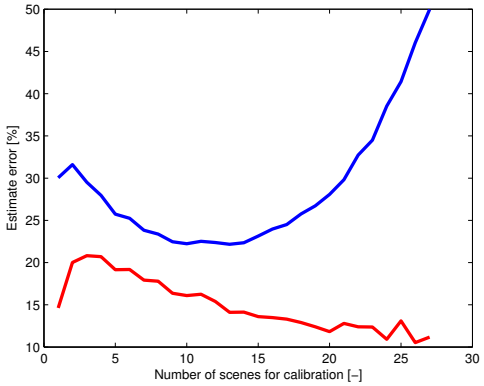


Figure 2: Estimate error for the rays from path tracing calibrated on rays from path tracing.

5 CONCLUSION AND FUTURE WORK

We have proposed an algorithm that combines a uniform grid and a hierarchical data structure for ray tracing so that it takes advantages of both types. Based on the scene properties and a small number of rays computed using the grid we decide either to continue ray tracing with the grid or to build the hierarchical data structure such as kd-trees.

We show that the use of uniform grids is relatively limited for standard scenes with the exception of scenes with a special distribution of geometric primitives in space. To our best knowledge we present the first algorithm that decides when it is advantageous to use uniform grids in dependence on the number of rays to be shot. Compared to the use of only a hierarchical data structure the method has a slowdown of only $1 + 1/log_2 N$ for the building of the data structure in the worst case. We can reach the average speedup 28.07 for a small number of rays. Our method can be used

Figure 3: All used test scenes. There are multiple scenes with the same model which differ only in a polygon count. Only one picture for each such group is shown. Scenes from the top-left are: a10, boxes, building, camel, bunny, sockets, case, conference, teapots, hunger, cornell, interior-3, interior-dance, interior-deloix, interior-japan, knot, teapot, sphere, phone, pills, chess, spheres.

in any hardware platform and any implementation of ray tracing that uses uniform grid and hierarchical data structure. We show that the number of rays that gives the critical point for the same performance of grids and hierarchical data structure can be well estimated with only a low number of scenes.

As a future work we can improve on estimates for the time needed to shoot a ray using yet unbuilt hierarchical data structure and the time to build this data structure for example by using other statistical characteristics of the scene. This could provide more accurate estimate for the critical point.

					Random rays			Path tracing		
Scene	Primitives	σ	T_B^G	T_B^H	t_T^G	t_T^H	R_C	t_T^G	t_T^H	R_C
boxes	528	7.46	2	12	5.0	4.1	11,669	0.9	1.4	0
interior dance	1,990	10.84	3	50	5.0	4.2	55,328	0.7	1.3	0
cornell	2,450	8.73	3	42	7.9	5.5	15,773	3.0	3.0	38,177
sphere	2,880	4.85	4	74	7.7	7.9	0	0.6	1.6	0
teapot	3,080	7.50	32	73	6.8	6.0	78,320	0.8	1.6	0
interior 3	3,412	10.75	4	83	4.6	3.8	101,888	0.9	1.0	124,367
phone	7,716	15.85	6	207	3.3	2.9	441,922	3.3	0.9	53,984
spheres	31,460	7.70	22	696	11.2	8.1	211,736	2.0	4.9	0
pills	32,606	12.90	21	802	7.1	3.6	221,538	2.3	1.8	593,553
teapots	52,360	12.43	29	1,215	13.4	6.8	172,843	4.3	5.4	1,620,297
building	54,490	42.19	21	995	6.3	3.1	304,414	12.8	1.8	63,055
knot	56,448	6.97	35	1,300	20.2	12.1	147,560	1.3	3.3	0
bunny	69,473	7.48	35	1,639	33.1	10.8	69,587	1.3	2.5	0
interior japan	72,310	43.03	36	1,241	4.6	4.8	0	3.1	1.9	515,863
sphere-high	87,120	6.68	48	1,751	23.5	13.5	161,984	1.1	2.8	0
case	131,228	24.93	51	2,872	14.0	4.2	293,541	5.3	3.0	638,451
hunger	141,143	64.21	47	2,586	15.9	5.6	250,041	68.2	3.1	236,153
interior deloix	149,090	16.02	67	3,928	22.9	6.6	231,927	3.0	1.9	1,704,493
camel	178,102	72.49	73	3,734	23.0	6.0	216,668	3.3	2.5	1,734,172
sockets	187,330	15.93	102	5,229	22.8	8.9	358,906	1.2	2.8	0
conference	190,947	25.62	97	4,428	19.4	7.1	348,928	2.8	2.7	3,268,361
chess	249,608	12.65	103	5,799	16.4	5.1	543,065	1.4	1.1	6,037,314
phone-high	318,756	30.72	145	8,163	6.7	2.9	2,262,962	7.5	1.5	856,324
pills-high	590,626	15.23	283	18,694	17.7	4.3	1,401,905	4.6	2.9	5,082,077
a10	1,070,671	60.79	403	29,921	25.0	5.9	1,544,378	2.6	3.1	52,604,130
hunger-high	1,418,560	136.61	384	27,690	44.2	5.6	727,590	156.3	3.9	27,356
case-high	1,614,006	36.86	501	37,224	39.6	4.5	1,108,126	11.1	4.5	3,101,617
sockets-high	1,658,432	16.80	730	45,842	41.0	11.0	1,516,267	2.7	4.1	0

Table 1: Measurements for all tested scenes for both ray generation schemes. T_B^G and T_B^H are uniform grid and kd-tree build times. t_T^G and t_T^H are uniform grid and kd-tree per-ray traversal times. Build times are in milliseconds and traversal times are in microseconds. R_C (the critical point) is the exact number of rays for which the uniform grid is equal in performance of the kd-tree.

ACKNOWLEDGEMENTS

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under research programs MSM 6840770014, LC-06008 (Center for Computer Graphics), MEB-060906 (Kontakt OE/CZ), the Grant Agency of the Czech Republic under research program P202/11/1883, and the Grant Agency of the Czech Technical University in Prague, grant No. SGS10/289/OHK3/3T/13.

REFERENCES

[App68] Appel, A. Some techniques for shading machine renderings of solids. In *AFIPS '68 (Spring): Proceedings of the April 30–May 2, 1968, spring joint computer conference*, pages 37–45, New York, NY, USA, 1968. ACM.

[Ben75] Bentley, J.L. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18:509–517, 1975.

[Cha04] Chang, A. Y.-H. *Theoretical and Experimental Aspects of Ray Shooting*. PhD thesis, Polytechnic University, USA, 2004.

[Cle88] Cleary, J.G. and Wyvill, G. Analysis of an algorithm for fast ray tracing using uniform space subdivision. *The Visual Computer*, 4(2):65–83, July 1988.

[Fuj86] Fujimoto, A., Tanaka, T., and Iwata, K. ARTS:

Accelerated ray tracing system. *IEEE Computer Graphics and Applications*, 6(4):16–26, 1986.

[Gla89] Glassner, Andrew S. An introduction to ray tracing, Academic Press Ltd.,London, UK, 1989

[Hav00a] Havran, V. *Heuristic Ray Shooting Algorithms*. Ph.d. thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000.

[Hav00b] Havran, V., Prikryl, J., and Purgathofer, W. Statistical comparison of ray-shooting efficiency schemes. Technical Report TR-186-2-00-14, Vienna University of Technology, May 2000.

[Hav07] Havran, V. About the relation between spatial subdivisions and object hierarchies used in ray tracing. In Mateu Sbert, editor, *23rd Spring Conference on Computer Graphics (SCCG 2007)*, pages 55–60, Budmerice, Slovakia, May 2007. ACM.

[Ize07] Ize, T., Shirley, P., and Parker, S. Grid creation strategies for efficient ray tracing. In *RT '07: Proceedings of the 2007 IEEE Symposium on Interactive Ray Tracing*, pages 27–32, Washington, DC, USA, 2007. IEEE Computer Society.

[Kal09] Kalojanov, J., and Slusallek, P. A parallel algorithm for construction of uniform grids. In *HPG '09: Proceedings of the Conference on High Performance Graphics 2009*, pages 23–28, New York, NY, USA, 2009. ACM.

[Kay86] Kay, T.L. and Kajiya, J.T. Ray tracing complex scenes. In David C. Evans and Rusell J. Athay, editors, *SIGGRAPH '86 Proceedings)*, volume 20, pages 269–278, August 1986.

[Kli94] Klimaszewski, K.S. *Faster ray tracing using adaptive grids and area sampling*. PhD thesis, Brigham Young University, dec 1994.

[MF99] Müller G. and Fellner D.W. Hybrid Scene Structuring with Application to Ray Tracing. In *Proceedings. of Intl. Conf. on Visual Computing ICVC '99*, 1999.

[SKH02] Szirmay-Kalos, L., Havran, V., Balázs, B. and Szécsi, L. On the efficiency of ray-shooting acceleration schemes. In *Proceedings of the 18th Spring Conference on Computer Graphics (SCCG 2002)*, pages 89–98, Budmerice, Slovakia, May 2002.

[Wal04] Wald, I. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004.

[Wal06a] Wald, I. and Havran, V. On building fast kd-trees for ray tracing, and on doing that in O(N log N). In *Proc. IEEE Symposium on Interactive Ray Tracing 2006*, pages 61–69, September 2006.

[Wal06b] Wald, I., Ize, T., Kensler, A., Knoll, A. and Parker, S.G. Ray Tracing Animated Scenes using Coherent Grid Traversal. *ACM Transactions on Graphics*, pages 485–493, 2006. (Proceedings of ACM SIGGRAPH 2006).

Biquadratic S-Patch in Bézier form

Alexej Kolcun
Institute of Geonics, Czech Academy of Sciences
Studentská 1768
708 00 Ostrava, Czech Republic
alexej.kolcun@ugn.cas.cz

ABSTRACT

Mutual conversions between triangular and quadrilateral meshes need the same degree of both diagonal and boundary curves of quadrilateral meshes. New approach to quadrilateral patches, S-Patches, offers such possibility. The Bézier approach of Smart patches (S-Patch) in the biquadratic case is analyzed. Dependencies among the control points are derived. BS-Patches are presented. Close relation between Bézier triangles and BS-Patches is found. Condition for smooth concatenation of biquadratic BS-Patches is formulated.

Keywords

Parametric modeling, S-Patch, Bézier patch, Bézier triangle.

1. INTRODUCTION

Two types of meshes, triangular and quadrilateral are used very often in various fields of computer graphic modeling [Pup_11]. Mutual conversions between them are the aim of interest for a long time, e.g. [Bru_80], [Far_86], [Gol_87], [Far_88]. Due to different geometric properties and incompatibility in these two types of meshes, it is difficult to use both kinds of patches in the same CAD system. Approximation techniques of the meshes mutual substitution are analyzed e.g. in [Lai_99]. Conversion of triangular patch to three quadrilateral ones is analyzed in [Hu_96]. Idea of degenerated rectangular meshes is used in [Hu_01]. Functional composition of the meshes is studied in [Fen_99] and [Las_02].

In [Hol_99] some properties of diagonal curve of the quadrilateral patch are analyzed. Importance of the main diagonal curves is recognized in [Ska_10], where the concept of Smart-Patches (S-Patch) is introduced. Here the main idea is to find suitable conditions, when both diagonal and boundary curves are the parametric curves of the same degree. It gives us a possibility to find simple and direct correlation between triangular and quadrilateral patches. In [Ska_10] bicubic patches in Hermit polynomial basis are analyzed.

In our approach we inspired with the idea mentioned above. We prefer Bernstein-Bézier form of polynomial basis functions. It is more convenient, due to the fact that we obtain the same formal description of both triangular and quadrilateral patches.

In this paper only the biquadratic case of S-Patches is analyzed in detail. (The importance and usefulness of biquadratic quadrilateral patches and quadratic triangular patches can be found e.g. in [Raz_05], [Boc_09].) Proves of main properties are presented in a very detailed way due to the fact, that in similar way the analysis of the patches of higher degree can be realized.

The rest of the paper is organized as follows. In section 2 biquadratic S-Patch is introduced in general form of simple polynomial basis functions (1, u, u^2). It gives us a basic form of S-Patch. In section 3 Bernstein-Bézier polynomial basis is used. Mutual dependencies of control points are analyzed. In section 4 it is shown when diagonal curves of S-Patch can be expressed as Bézier curves of proper 'diagonal' control points. Such patches are introduced as BS-Patches. In section 5 it is shown, that BS-Patches we can split to Bézier triangle patches. In section 6 conditions of smooth concatenation of BS-Patches are formulated.

2. PROBLEM FORMULATION

Let us consider biquadratic parametric patch

$$X(u,v)=\mathbf{uRv}^T=\begin{pmatrix}1 & u & u^2\end{pmatrix}\begin{pmatrix}R_{00} & R_{01} & R_{02}\\ R_{10} & R_{11} & R_{12}\\ R_{20} & R_{21} & R_{22}\end{pmatrix}\begin{pmatrix}1\\ v\\ v^2\end{pmatrix}^T \quad (1)$$

Our goal is to find out the conditions for all boundary lines and both main diagonals $D_1(u)$, $D_2(u)$ to be the lines of the same degree.

$$D_1(u) = X(u,u) = \mathbf{u}\,\mathbf{R}\,\mathbf{u}^T = A_0 + A_1 u + A_2 u^2 \qquad (2)$$

$$D_2(u) = X(u,1-u) = \mathbf{u}\mathbf{R}(\mathbf{1}-\mathbf{u})^T = \\ = \mathbf{u}\mathbf{R}\begin{pmatrix}1 & 0 & 0\\ 1 & -1 & 0\\ 1 & -2 & 1\end{pmatrix}\mathbf{u}^T = B_0 + B_1 u + B_2 u^2 \qquad (3)$$

Such patches are named S-Patches [Ska_10].

Theorem 1. Biquadratic patch (1) is a S-Patch iff

$$R_{12} = R_{21} = R_{22} = 0,$$

i.e.

$$X(u,v) = \mathbf{u}\begin{pmatrix}R_{00} & R_{01} & R_{02}\\ R_{10} & R_{11} & 0\\ R_{20} & 0 & 0\end{pmatrix}\mathbf{v}^T \qquad (4)$$

Proof: Resulting matrix for (3) is

$$\begin{pmatrix}R_{00}+R_{01}+R_{02} & -R_{01}-2R_{02} & R_{02}\\ R_{10}+R_{11}+R_{12} & -R_{11}-2R_{12} & R_{12}\\ R_{20}+R_{21}+R_{22} & -R_{21}-2R_{22} & R_{22}\end{pmatrix}$$

So, the conditions (2) and (3) lead to the equations

$$\begin{aligned} R_{12} + R_{21} &= 0\\ R_{22} &= 0\\ R_{12} - R_{21} - 2R_{22} &= 0\\ R_{22} &= 0 \end{aligned} \qquad (5)$$

It is obvious, that linear system (5) has trivial solution only

$$R_{12} = R_{21} = R_{22} = 0.$$

QED.

Corollary. All parametric lines of a biquadratic S-Patch are curves of degree $d \le 2$.

Proof: Let us consider general parametric line of S-patch. Using standard transformations of (4) for $L(u) = X(u, a+bu)$ we obtain the resulting formula

$$L(u) = \mathbf{u}\begin{pmatrix}Q_{00} & Q_{01} & Q_{02}\\ Q_{10} & Q_{11} & 0\\ Q_{20} & 0 & 0\end{pmatrix}\mathbf{u}^T$$

where

$$\begin{aligned} Q_{00} &= R_{00} + aR_{01} + a^2 R_{02},\\ Q_{01} &= bR_{01} + 2abR_{02},\\ Q_{10} &= R_{10} + aR_{11},\\ Q_{02} &= b^2 R_{02},\\ Q_{11} &= bR_{11},\\ Q_{20} &= R_{20}. \end{aligned}$$

Q.E.D.

3. BÉZIER FORM OF S-PATCH

Let us express the biquadratic S-Patch in Bézier form

$$X(u,v) = \mathbf{u}\mathbf{R}\mathbf{v}^T = \mathbf{u}\begin{pmatrix}1 & 0 & 0\\ -2 & 2 & 0\\ 1 & -2 & 1\end{pmatrix}\mathbf{P}\begin{pmatrix}1 & -2 & 1\\ 0 & 2 & -2\\ 0 & 0 & 1\end{pmatrix}\mathbf{v}^T \qquad (6)$$

From (6) we can find control points P_{ij}.

$$\mathbf{P} = \begin{pmatrix}1 & 0 & 0\\ -2 & 2 & 0\\ 1 & -2 & 1\end{pmatrix}^{-1}\mathbf{R}\begin{pmatrix}1 & -2 & 1\\ 0 & 2 & -2\\ 0 & 0 & 1\end{pmatrix}^{-1} \qquad (7)$$

In more detailed way

$$\begin{pmatrix}P_{00} & P_{01} & P_{02}\\ P_{10} & P_{11} & P_{12}\\ P_{20} & P_{21} & P_{22}\end{pmatrix} = \begin{pmatrix}1 & 0 & 0\\ 1 & \frac{1}{2} & 0\\ 1 & 1 & 1\end{pmatrix}\begin{pmatrix}R_{00} & R_{01} & R_{02}\\ R_{10} & R_{11} & 0\\ R_{20} & 0 & 0\end{pmatrix}\begin{pmatrix}1 & 1 & 1\\ 0 & \frac{1}{2} & 1\\ 0 & 0 & 1\end{pmatrix}$$

Explicit vector form of (7) gives

$$\begin{pmatrix}P_{00}\\ P_{01}\\ P_{02}\\ P_{10}\\ P_{11}\\ P_{12}\\ P_{20}\\ P_{21}\\ P_{22}\end{pmatrix} = \frac{1}{4}\begin{pmatrix}4 & 0 & 0 & 0 & 0 & 0\\ 4 & 2 & 0 & 0 & 0 & 0\\ 4 & 4 & 0 & 4 & 0 & 0\\ 4 & 0 & 2 & 0 & 0 & 0\\ 4 & 2 & 2 & 0 & 0 & 1\\ 4 & 4 & 2 & 4 & 0 & 2\\ 4 & 0 & 4 & 0 & 4 & 0\\ 4 & 2 & 4 & 0 & 4 & 2\\ 4 & 4 & 4 & 4 & 4 & 4\end{pmatrix}\begin{pmatrix}R_{00}\\ R_{01}\\ R_{10}\\ R_{02}\\ R_{20}\\ R_{11}\end{pmatrix} \qquad (8)$$

Rank of the matrix **M** in (8), *rank*(**M**) = 6.

In the text below (Figures 2 – 4) we use vector indexing and Cartesian indexing of control points of a patch – Fig. 1.

1	2	3	00	01	02
4	5	6	10	11	12
7	8	9	20	21	22
a)			b)		

Figure 1. a) vector indexing, b) Cartesian indexing of control points.

The first six rows of matrix **M** in (8) (Fig. 2a)) are linearly dependent, as we can write

$$P_{12} = P_{00} - 2P_{01} + P_{02} - P_{10} + 2P_{11}.$$

Similarly, the sets of rows in (8) (i.e. the rows of matrix **M**) (1,2,3,7,8,9), (4,5,6,7,8,9), (1,4,7,2,5,8), (1,4,7,3,6,9), (2,5,8,3,6,9) are linearly dependent too.

a) b) c)

Figure 2. Configurations of dependent 6-element sets of control points P_{ij} (black).

For the configuration of points Fig. 2c), i.e. for the configuration of rows (1,2,4,6,8,9) of the matrix **M** in (8) we can find relation

$$P_{22} = P_{12} + P_{21} - P_{01} - P_{10} + P_{00}.$$

The symmetric configuration of rows (2,3,4,6,7,8) is linearly dependent too.

We can formulate the condition for the independency of sets of the control points P_{ij}.

Theorem 2. Only the eight six-element sets of control points mentioned above are linearly dependent.

Proof can be done by computing the determinant of all $\binom{9}{6} = 84$ 6x6 submatrices of **M** in (8).

(Due to symmetries it is enough to parse not more than 21 cases.)

QED.

Such configurations of control points cannot be used for the patch determination.

Fig. 3 gives examples of independent sets of control points. Symmetrical cases are independent too.

a) b) c) d)

Figure 3. Some configurations of independent 6-element sets of control points (black).

Useful properties of some independent configurations of control points are:

1. configurations from Fig. 3a), 3b) involve all corner control points,
2. configurations from Fig. 3a), 3c) involve full information of the pair of neighbour boundary lines.

4. BS-PATCH

Let us analyze the relations between main diagonal $D_1(u)$ of S-Patch (4)

$$D_1(u) = \mathbf{u}\begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & R_{11} & 0 \\ R_{20} & 0 & 0 \end{pmatrix}\mathbf{u}^T \tag{9}$$

and proper Bézier diagonal – i.e. the curve defined on the set of 'diagonal' control points P_{00}, P_{11}, P_{22} – Bézier diagonal curve

$$D_{1B}(u) = \mathbf{u}\begin{pmatrix} 1 & 0 & 0 \\ -2 & 2 & 0 \\ 1 & -2 & 1 \end{pmatrix}\begin{pmatrix} P_{00} \\ P_{11} \\ P_{22} \end{pmatrix}. \tag{10}$$

Using the abbreviation

$$\Re = (R_{00} \quad R_{01} \quad R_{10} \quad R_{02} \quad R_{20} \quad R_{11})^T,$$

for (9) we obtain

$$D_1(u) = \mathbf{u}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix}\Re \tag{11}$$

On the other hand, for Bézier diagonal curve the resulting expression of (10) is

$$D_{1B}(u) = \mathbf{u}\begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \end{pmatrix}\Re. \tag{12}$$

As the matrices in (11) and (12) differ in the last column only, the condition $R_{11} = 0$ must be fulfilled for both Bézier and S-patch diagonals to be identical.

The same result we obtain for the diagonal curves $D_2(u)$ and $D_{2B}(u)$.

$$D_2(u) = \mathbf{u}\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -2 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{pmatrix}\Re, \tag{13}$$

$$D_{2B}(u) = \mathbf{u}\begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 1 & -2 & 0 & 0.5 \\ 0 & 0 & 0 & 1 & 1 & 0.5 \end{pmatrix}\Re. \tag{14}$$

Here the matrices in (13) and (14) also differ in the last column only.

Just proved relations among the diagonal lines can be formulated as the theorem below.

a) b) c) d) e) f) g)

Figure 4. 5-element sets of control-points. a),b) – non independent, c)–g) independent sets.

Theorem 3. $D_1(u)= D_{1B}(u)$ if and only if $R_{11}=0$. Moreover, equality of these diagonals automatically implies the equality of $D_2(u)= D_{2B}(u)$.

On the base of the Theorem 3 we can introduce biquadratic BS-Patch, i.e. patch in the form as follows

$$X(u,v)=\mathbf{u}\begin{pmatrix} R_{00} & R_{01} & R_{02} \\ R_{10} & 0 & 0 \\ R_{20} & 0 & 0 \end{pmatrix}\begin{pmatrix} 1 \\ v \\ v^2 \end{pmatrix}\mathbf{v}^T . \qquad (15)$$

In this case mutual relations among Bézier control points P_{ij} (8) are reduced to

$$\begin{pmatrix} P_{00} \\ P_{01} \\ P_{02} \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{20} \\ P_{21} \\ P_{22} \end{pmatrix} = \frac{1}{2}\begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 & 0 \\ 2 & 2 & 0 & 2 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 2 & 0 \\ 2 & 0 & 2 & 0 & 2 \\ 2 & 1 & 2 & 0 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix}\begin{pmatrix} R_{00} \\ R_{01} \\ R_{10} \\ R_{02} \\ R_{20} \end{pmatrix}. \qquad (16)$$

We can see that now the corner control points (Fig. 4a)) are dependent,

$$P_{22} = P_{20} + P_{02} - P_{00} .$$

It means that the corner control points create rhomboids.

Similarly the quaternion of neighbour control points (Fig. 4b)) is dependent too,

$$P_{11} = P_{10} + P_{01} - P_{00} .$$

Examples of non independent and independent 5-element sets of control points of BS-Patches are presented in Fig. 4 c) – g). E.g. for independent pentad from Fig. 4e) the rest of control points can be represented as

$$\begin{aligned} P_{00} &= P_{01} + P_{10} - P_{11} \\ P_{02} &= P_{01} + P_{12} - P_{11} \\ P_{20} &= P_{21} + P_{10} - P_{11} \\ P_{22} &= P_{21} + P_{12} - P_{11} \end{aligned} . \qquad (17)$$

5. BS-PATCH AND BÉZIER TRIANGLES

As both diagonal and boundary curves of BS-Patches are Bézier curves, it is meaningful to analyze the triangle patches. We shall demonstrate that there is a very close connection between the Cartesian BS-Patch and a pair of triangular Bézier patches. This relation is formulated for the case n =2.

Let us consider triangular mesh of nodes

$$P_{ijk} \quad 0 \le i, j, k \le n, \quad i + j + k = n$$

where nodes $P_{i_1 j_1 k_1}$, $P_{i_2 j_2 k_2}$ are neighbour, if

$$|i_1 - i_2| + |j_1 - j_2| + |k_1 - k_2| = 2 .$$

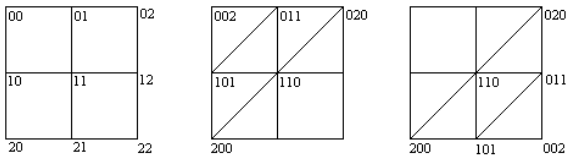
Bézier triangular patch is defined as

$$P_\Delta(u,v,w,\{P_{ijk}\}) = \sum_{(i,j,k)} \frac{n!}{i!\,j!k!} u^i v^j w^k P_{ijk} \qquad (18)$$

where

$$0 \le u,v,w \le 1,\ u+v+w=1,\ 0 \le i, j, k \le n,\ i + j + k = n .$$

Let us consider quadratic BS-Patch defined on the set of control points $P_{ij}, 0 \le i, j \le 2$. Let us consider Cartesian and triangular indexing of these control points according to Fig. 5.



a) b) c)

Figure 5. Cartesian a) and triangular b), c) indexing of control nodes for n = 2.

Theorem 4. BS-Patch defined on control points $P_{ij}, 0 \le i, j \le 2$ is the same surface as the pair of triangular Bézier patches, defined on the sets of proper control points.

Proof. Let us use the independent set of control points according to Fig. 4e). Solving proper subsystem of (16)

$$\begin{pmatrix} P_{01} \\ P_{10} \\ P_{11} \\ P_{12} \\ P_{21} \end{pmatrix} = \frac{1}{2} \begin{pmatrix} 2 & 1 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 0 \\ 2 & 1 & 1 & 0 & 0 \\ 2 & 2 & 1 & 2 & 0 \\ 2 & 1 & 2 & 0 & 2 \end{pmatrix} \begin{pmatrix} R_{00} \\ R_{01} \\ R_{10} \\ R_{02} \\ R_{20} \end{pmatrix}$$

and inserting the solution into (15) we obtain

$$\mathbf{R} = \begin{pmatrix} P_{01} + P_{10} - P_{11} & 2(P_{11} - P_{10}) & P_{10} + P_{12} - 2P_{11} \\ 2(P_{11} - P_{01}) & 0 & 0 \\ P_{01} + P_{21} - 2P_{11} & 0 & 0 \end{pmatrix}. \quad (19)$$

Let us express triangular patch on the nodes

$P_{00}, P_{01}, P_{02}, P_{10}, P_{11}, P_{20}$ – Fig. 5a).

Using the triangular indexing – Fig. 5b), we have

$$\begin{aligned} & P_{\Delta}(u,v,w) = \\ & = u^2 P_{200} + v^2 P_{020} + w^2 P_{002} + \\ & \quad + 2uvP_{110} + 2uwP_{101} + 2vwP_{011} \end{aligned} \quad (20)$$

Rewriting it to Cartesian indexes – Fig. 5a) we obtain

$$\begin{aligned} & P_{\Delta}(u,v,w) = \\ & = u^2 P_{20} + v^2 P_{02} + w^2 P_{00} + \\ & \quad + 2uvP_{11} + 2uwP_{10} + 2vwP_{01} \end{aligned}$$

Using (17) and excluding *w*, as $w = 1 - u - v$ leads to the final form of triangle patch

$$\begin{aligned} & P_{\Delta}(u,v,w) = P_{01} + P_{10} - P_{11} + \\ & + 2u(P_{11} - P_{01}) + 2v(P_{11} - P_{10}) + \\ & + u^2(P_{01} + P_{21} - 2P_{11}) + v^2(P_{10} + P_{12} - 2P_{11}). \end{aligned}$$

We have obtained the same relation as (19).

For the triangle defined on control points form Fig. 5c) the process of proving is similar. Difference is in used parametrization only: in (20) we use $(1-u)$ instead of *u* and $(1-v)$ instead of *v*.

For triangles with the diagonal defined on control points P_{00}, P_{11}, P_{22} in (20) we have to use parametrizations $(1-u), v$, $u, (1-v)$ respectively.

QED.

Corollary. Just proved theorem gives us an important generalizaton of the trivial fact that a bilinear patch can be decomposed to two triangles iff the quaternion of control points is planar.

6. SMOOTH CONCATENATION OF BS-PATCHES

Let us consider 5-element set of independent control points form Fig. 4e). Condition (17) says that the set of control points creates four rhomboids – Fig. 6. Here we can distinguish three types of control points: 'central', 'crosswise' and 'dependent'.

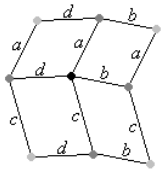
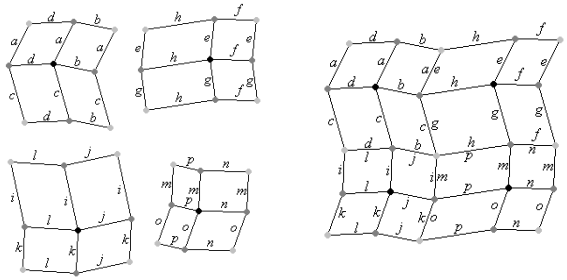


Figure 6. Resulting geometry of control points for BS-patch. Different types of control points are distinguished: black – central one, dark – crosswise ones, light – dependent ones.

Let us consider four general BS-patches – Fig. 7a). The conditions for concatenation of the patches are obvious – Fig 7b):

a=e, c=g, i=m, k=o, d=l, b=j, h=p, f=n.

This condition can be formulated more generally in the following way.



a) b)

Figure 7. Concatenation of BS-patches.

a) Four independent BS-patches. b) Concatenated BS-patches.

Let there are two open polylines

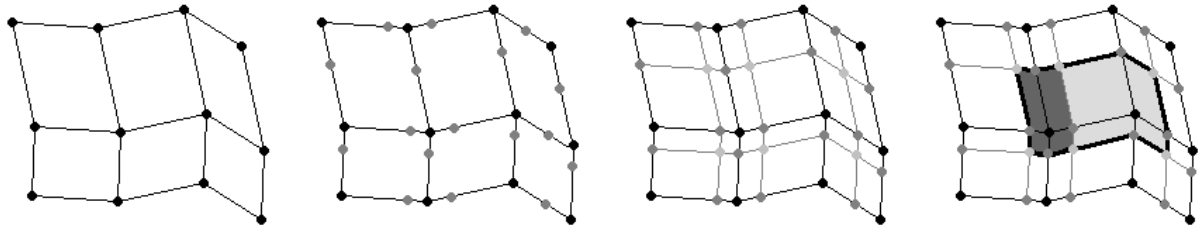
$\Lambda_1 = (P_0 P_1 P_2 \cdots P_n)$ and $\Lambda_2 = (R_0 R_1 R_2 \cdots R_m)$.

Let us consider the lattice of nodes

$$\Lambda = (Q_{i,j} : 0 \le i \le n, 0 \le j \le m)$$

where

$$x_{i,j} = x_{P_i} + x_{R_j} - x_{R_0}, \quad y_{i,j} = y_{R_j} + y_{P_i} - x_{P_0}.$$



a) b) c) d)

Figure 8. Smooth concatenation of BS-patches according to the steps a) – d) below.

Theorem 5. Surface is set of BS-Patches iff set of control points is a lattice of polylines.

Moreover, if we demand smooth concatenation of BS-Patches, edges *b,h* must be parallel. Similarly, edges *c,i* must be parallel too. It means that the quaternion of central control points from Fig. 7b) creates the vertices of rhomboid.

Construction

Given two polylines

$$\Lambda_1 = (P_0 P_1 \cdots P_n),\ \Lambda_2 = (R_0 R_1 \cdots R_m),$$

given two sets

$$\pi = (p_0, p_1, \cdots, p_{n-1}), \rho = (r_0, r_1, \cdots, r_{m-1}), 0 < p_i, r_j < 1,$$

we can construct smooth concatenation of BS-patches according to the steps below.

a) We suppose that the central control points of BS-patches create a lattice.
b) Crosswise control points can be found as a ratio of neighbour central control points.
c) Dependent control points (corners of BS-patches) are found according to the (17).
d) Concatenation consists of full-defined BC-patches.

Fig. 8 illustrates the above described construction.

7. CONCLUSIONS

In the presented study we have described the Bézier form of S-Patches in the biquadratic case.

- Dependencies among the control points are derived.
- BS-Patches are introduced.
- Close relation between Bézier triangles and BS-Patches is found.
- Condition for smooth concatenation of biquadratic BS-Patches is formulated.

We can see that biquadratic BS-patches are very convenient for mutual conversion between triangular and quadrilateral patches. On the other hand, smooth concatenation of such patches is 'too rigid' and perhaps it is hardly used for the shape expression in general case. Future work will be focused to

- more detailed analysis of relationship between S-Patches and BS-Patches,
- patches of higher degree.

8. ACKNOWLEDGMENTS

This work is supported by the grant GACR 105/09/1830 of the Grant Agency CR and the research plan AVOZ 30860518 of the Academy of Sciences of the Czech Republic.

REFERENCES

[Boc_09] Bocek, J., Kolcun, A.: Shading of Bézier patches, in: GraVisMa 2009 workshop proc. (V. Skala, D. Hildenbrand eds.) Univ. of West Bohemia, Plzeň, 2009, pp. 126-129.

[Bru_80] Brueckner, I.: Construction of Bézier points of quadrilateral forms those of triangles, Computer-Aided Design 12,1(1980), pp. 21-24.

[Fen_99] Feng, J.Q., Peng, Q.S.: Functional Compositions via Shifting Operators for Bézier Patches and Their Applications, Chinese Journal of Advaced Software Research, 10(1999), pp. 1316-1321.

[Far_86] Farin, G.E.: Triangular Bernstein-Bézier patches, CAGD 3,2(1986), pp 83-127.

[Far_88] Farin, G.E.: Curves and Surfaces for CAGD: A practical Guide, Academic Press, 1988.

[Gol_87] Goldman, R.N., Filip, D.J.: Conversion from Bézier rectangles to Bézier triangles, Computer-Aided Design, 19,1(1987), pp. 25-27.

[Hol_99] Holliday D.J., Farin, G.E.: A geometric interpretation of the diagonal of a tensor-product Bézier volume, CAGD 16(1999) pp. 837-840.

[Hu_96] Hu, S.M.: Conversion of a triangular Bézier patch into three rectangular Bézier patches, CAGD 18(2001) pp. 219-226.

[Hu_01] Hu, S.H.: Conversion between triangular and rectangular Bézier patches, CAGD 18(2001) pp. 667-671.

[Lai_99] Lai, M.J., Schumaker, L.L.: On the Approximation Power of Splines on Triangulated Quadrangulations, SIAM, 36,1(1999), pp.143-159)

[Las_02] Lasser, D.: Tensor product Bézier surfaces on triangle Bézier surfaces, CAGD 19(2002) pp. 625-643.

[Pup_11] Puppo E.: Quad meshes vs Triangle meshes: What is better?, Keynote lecture, WSCG 2011. http://wscg.zcu.cz/WSCG2011/Papers_2011/Puppo.htm

[Raz_05] Razdan, A., Bae, M.S.: Curvature Estimation Scheme for Triangle Meshes Using Biquadratic Bézier Patches, Computer-Aided Design 37,14(2005) pp. 1481-1491.

[Ska_10] Skala, V., Ondračka, V.: S-Patch: Modification of the Hermite parametric patch, in Conf.proc. ICGG 2010, Kyoto 2010.